
**Discrete Event Simulation as a Risk Analysis Tool for
Remote Afterloading Brachytherapy**

Prepared by Rick Archer, John Keller, Sue Archer,
and Shelly Scott-Nash

Micro Analysis & Design Inc.
4900 Pearl East Circle, Suite 201
Boulder, CO 80301

Paul M. Lewis
NRC Project Manager

Prepared for
Office of Nuclear Regulatory Research

U.S. Nuclear Regulatory Commission
Washington, DC 20555-0001

ABSTRACT

Remote Afterloading Brachytherapy (RAB) is a cancer treatment process that uses ionizing radiation from radioactive materials to retard or destroy tumors. RAB uses a computer-controlled device to insert radioactive sources in or adjacent to the tissue to be exposed. Human errors and equipment failures in the RAB process can lead to the administration of incorrect radiation doses, the insertion of radioactive sources into the incorrect location in the body, or a treatment delivered to the wrong patient.

Discrete Event Simulation (DES) is a technique for simulating and studying processes that can be described by a sequence of events that each have a distinct beginning and end. It is particularly well suited to modeling series of human tasks. This study used DES modeling to evaluate the potential for human errors in the process and responses made by humans to hardware and software failures. Using a task analysis of RAB combined with estimates of the probability of the errors that can occur in the RAB process, the events (previous tasks and errors) that lead to the errors, and information about potential equipment failures, a DES model of the process was developed. This project demonstrated that the DES model can be used to examine process safety through the identification of the human errors that are most likely to occur and the points in the process most vulnerable to severe consequences.

I

APPENDICES

(See Volume 2)

FIGURES

1. Example of a task network diagram	7
2. The Micro Saint user interface	16
3. Schematic example of a task network	16
4. Task Description dialog box	17
5. Decision Description dialog box	19
6. Variable Description dialog box	20
7. Event Description dialog box	22
8. Snapshot Description dialog box	23
9. The arrow between the two functions in the RAB model showing the order of execution	32
10. Task Description dialog box for the <i>Count Applicators</i> task	32
11. Addition of <i>Collect Data</i> and <i>Next Patient</i> functions	33
12. Example of a network with a branch of three paths	34
13. Decision Description dialog box for a three-way branch	34
14. Example of a network with a Tactical decision type	34
15. Decision Description dialog box for Tactical decision	35
16. Task diagram for multiple patients	35
17. Task diagram showing connection between nodes	36
18. Decision Description dialog box showing routing conditions for multiple patients	36

19. Path connection for the error initiation node	<u>38</u>
20. Path connections for multiple error initiated from a single task	<u>38</u>
21. Network diagram showing an error catch	<u>38</u>
22. Mitigation paths for two errors that could be caught	<u>39</u>
23. Example of a network diagram showing initiated and caught errors	<u>39</u>
24. Diagram showing initiation point for the single error	<u>41</u>
25. Error Initiation points for multiple errors	<u>42</u>
26. Error Catch for single error	<u>43</u>
27. Error Catch for multiple errors	<u>44</u>
28. Diagram showing error initiation and error catch points	<u>46</u>
29. Event Description dialog box	<u>48</u>
30. Snapshot Description dialog box	<u>50</u>
31. Error initiation and catch points	<u>65</u>
32. Patient preparation network diagram	<u>70</u>
33. Treatment delivery network diagram	<u>70</u>
34. Session monitoring network diagram	<u>71</u>

TABLES

1. Partial list of errors from the RAB model	28
2. New patient programming function code	36
3. Initialize Patient Variables programming function code	37
4. Error Initiation programming function code	40
5. Error Caught programming function code	41
6. Error Initiation expressions for a single error	42
7. Error Initiation expressions for multiple errors	43
8. Error Catch expressions for a single error	44
9. Error Catch expressions for multiple errors	45
10. Error initiation and catch point expressions	46
11. Ending Effect dependency code example	47
12. NUREG/CR-6125 vs. Micro Saint terminology	60

13. Function and task inventory for RAB	61
14. Quality Assurance error modifications	63
15. Error categories	67
16. Probability distributions of dosage deviation	68
17. Settings for the four scenario runs	77
18. Dosage deviations for all fractions	78
19. Dosage deviations for last fractions	78
20. Summary of most frequently occurring errors (table shows error numbers)	79
21. Descriptions for the most frequently occurring errors	79
22. Summary of tasks where the most errors were initiated (table shows task numbers)	80
23. Numbers and names for each task where the most errors were initiated	80

EXECUTIVE SUMMARY

Introduction

Remote Afterloading Brachytherapy (RAB) is a cancer treatment process that uses ionizing radiation from radioactive materials to retard or destroy tumors. RAB uses a computer-controlled device to remove the radioactive materials from a shielded container and move them through tubes into applicators that are placed in or adjacent to a tumor. The device controls the dwell positions and the dwell times inside the applicator. The device then returns the radioactive materials to the shielded container. Human errors and equipment failures in the RAB process can lead to the administration of incorrect radiation doses, the insertion of radioactive sources into the incorrect location in the body, or a treatment delivered to the wrong patient.

The Nuclear Regulatory Commission (NRC) regulates aspects of RAB related to radiation safety as part of its mission to protect public health and safety. In 1995, the NRC completed an evaluation of RAB designed to identify potential human errors, their causes and consequences and to prioritize function and task performance problems related to human errors in terms of their safety significance. The results of that study, published in, Human Factors Evaluation of Remote Afterloading Brachytherapy (NUREG/CR-6125) Vols. 1-3, 1995, is a primary source of information for this research.

The two primary objectives of the project documented in this report were (1) to determine the feasibility of using Discrete Event Simulation (DES) modeling to assess the effects on process safety of human errors and hardware and software failures in RAB in particular and in medical processes using radioactive materials in general, and (2) to compare DES with other risk assessment methods for assessing risks associated with medical processes using radioactive materials. To accomplish these objectives, a DES model was constructed for the RAB treatment process. Possible human errors and human responses to hardware and software failures were included in the model.

The NRC is currently developing guidelines and methods for implementing a policy called "Risk-Informed Regulation." For nuclear power plants, these guidelines rely heavily on traditional Probabilistic Risk Assessment (PRA) and Human Reliability Assessment (HRA) methods, most of which rely on traditional applications of fault trees and event trees. This project demonstrates that DES is an alternative risk assessment method that might be more appropriate than the traditional PRA/HRA methods for assessing risks associated with medical use of radioactive materials

Discrete Event Simulation

A simulation is an abstract representation or model of a system. Simulation helps the analyst gain a better understanding of the system. DES is used to study processes that can be described by a sequence of events that each have a distinct beginning and end. A computer is used to simulate execution of these events. DES has been successfully used to analyze aspects of processes, such as, human performance, resource utilization, efficiency, cost, scheduling, and training.

In DES, tasks are displayed schematically on a network diagram and are connected by arrows that show potential sequences of task execution. The possibility that more than one task can follow a specific task is indicated by multiple arrows coming from that task and pointing to multiple potential subsequent tasks. Probabilities or mathematical equations are used to determine the path(s) that should be followed. Any task or occurrence, such as, initiating an error, catching an error, or an equipment failure, can be represented in the network diagram.

Once the model is constructed, the simulation can be executed multiple times. Variables can be defined to record specific values that represent the status of the system as the model is executed. Statistical analysis of the recorded variables from multiple executions can identify the most frequent errors and their associated tasks. The specific areas for improving process safety can then be identified.

Once specific process changes are proposed, the model can be used to test how those changes affect the overall activity. The results can then be compared against original (baseline) results to see the effect the changes had on the process. This method can be repeated many times to obtain the most desirable result.

Methodology

In this study, a methodology was developed for using DES for risk assessment. A model of the RAB process was developed to demonstrate the methodology. The RAB model was constructed based on the human factors evaluation of RAB documented in NUREG/CR-6125 and the use of Nucletron's Microselectron High Dose Rate Remote Afterloading Brachytherapy equipment. The model considered the numbers and types of errors and equipment failures in the RAB process and the events that lead to incorrect treatments. The following is a list and short description of each of the steps used to create the RAB model. The list is also a list of typical steps and a guide that can be used in building similar models in the future.

1. Collect initial process information
The information used in the initial construction of the RAB process model was collected from several different sources. These included NUREG/CR-6125, interviews with Subject Matter Experts (SMEs), visits to treatment facilities, NRC occurrence reports, and the user's manual for the RAB equipment.
2. Build the initial network diagram
Building a network diagram is the first step performed using the Micro Saint simulation software. For this project, the network diagram was constructed to represent the RAB treatment process.
3. Develop and add error probability distributions
The information collected on potential RAB process errors was used to develop probability distributions for each error. These were added to the model to represent the initiation and the catching of errors.
4. Add error dependencies
The initiation of some errors can increase or decrease the likelihood of other errors. Information collected on the dependencies between some errors was used to demonstrate this capability of the model.
5. Add equipment failures
Potential RAB equipment (hardware and software) failures were included in the model in the form of failure scenarios. There was no attempt to represent all the equipment failures that could occur in the RAB process. The hardware failure and software error scenarios are meant to demonstrate one way these types of problems can be represented in a DES.
6. Define execution parameters
A simulation uses a set of parameters to control its execution. For the RAB model, some of these parameters included specifics of patient treatment, the number of patients, and the number of treatments per week.
7. Test and debug the model
Testing and debugging a computer model is an iterative process that takes place throughout the

model's construction. This step is listed at this point in the methodology to indicate that all the testing and debugging was completed prior to using the model for the purpose of analysis.

8. Verify and validate (V&V) the model

The steps of verifying and validating a model are iterative and can take place several times during the construction and use of the model. During this project, we used several different levels of V&V at several different points in the process.

9. Modify the model based on the V&V

Additional information was obtained during the final V&V. Several modifications were made to the model to reflect this information.

10. Execute the model

The model was executed several different times using different input values to simulate different scenarios of interest.

11. Collect model execution data

Each execution of the model produced data on the results of the scenario that was simulated. These results were used to determine the risks involved with the process of providing RAB treatments.

Verification and Validation

Two separate verification and validation (V&V) efforts were performed for this project. The first occurred after the basic RAB treatment process model was developed. The primary source of information for constructing the model was NUREG/CR-6125, Human Factors Evaluation of Remote Afterloading Brachytherapy Vols. 1-3, 1995. To verify that the model accurately represented the information in NUREG/CR-6125 and to validate that it was a reasonable representation of reality for performing RAB treatments, the project team met with the authors of NUREG/CR-6125 to review the DES model. Minor modifications to the model had to be made as a result of this review. For the most part, the review indicated that the RAB model did accurately represent the process documented in NUREG/CR-6125. The RAB processes documented in NUREG/CR-6125 were a composite of process steps performed at the 23 RAB facilities that were sampled in that study. As a result, the RAB process that was modeled did not represent the actual process at any particular facility. However, the DES techniques used in the model were easy to communicate and well understood by the subject matter experts who performed the human factors evaluation for NUREG/CR-6125. This made the model easy to review and evaluate.

The second V&V effort was conducted at a medical facility that routinely performs RAB treatments using the same equipment and many of the techniques included in the RAB DES model. In attendance at this review were medical physicists from two separate facilities that perform RAB treatments. The finding was that the processes for performing RAB treatments were considerably different at different facilities and also considerably different from one treatment type to another. From this, it was concluded that a generic process model for all RAB treatments at any facility was not an accurate way to represent reality. However, the subject matter expert reviewers felt that the DES model was easy to understand and evaluate. In addition, each of the experts indicated that DES seemed to be a useful way to represent RAB treatment processes and felt that they could provide the information required to modify the DES model to better represent the processes at their respective facilities with minimal effort.

Results

A sensitivity study was performed using two variables: (1) error initiation probabilities, which were varied between high and low, and (2) error dependencies, which were included or excluded. The four combinations derivable from the two values of these two variables constituted four scenarios. The number of patients for each scenario was set at 1,000. For each of the four scenarios, the most frequently occurring errors and associated tasks were identified. One of the most interesting results from the four model executions was that the same set of the most frequently occurring errors was found in all four scenarios. In addition, the same tasks were responsible for initiating the highest number of errors that resulted in incorrect treatments. The results of the study suggest that simulation can be used to identify areas of risk of incorrect treatment in the RAB process even if the probability of error data are relative instead of absolute. DES can be used to help identify the points in the process at which errors are most likely to occur, and the points where errors can be detected and corrected. DES can also be used to evaluate alternative solutions to the parts of the process that are sources of human error. The RAB model developed for this project is a composite of several different facilities. In order to analyze RAB at a specific facility, a facility-specific model should be developed. The ease with which subject matter experts could understand and review the DES model is a very important strength of DES. Furthermore, it may be easier for an experienced analyst to construct a DES model of RAB than for an experienced analyst to construct a fault tree or event tree analysis of the same process¹.

Limitations

Several limitations in the scope of the project and the model are as follows: The treatment of hardware and software failures is not meant to be a thorough treatment of all possible equipment failures within the RAB process. Rather, it is intended to show how equipment failures, their effects on the process, and human responses to the failures can be represented in a DES model. This model is not intended to be an accurate predictor of treatment outcomes for a specific facility because it does not represent the RAB process from a specific facility. As a result, the SMEs from one of the facilities visited were unable to provide a validation of the input data for the model (e.g., error initiation and catch probabilities), which were solicited from the authors of NUREG/CR-6125. However, these probabilities can be easily altered within the model to perform sensitivity analysis.

Conclusions

DES is an excellent technique for simulating processes that can be described by discrete events and can be effectively used to assess risk in medical procedures that use radioactive by-products. A DES model can also be used to answer other questions that address process safety. For example, the effects of proposed changes in the medical process can be assessed by comparing the results of the baseline model with the results of an alternative model of the proposed changes. This can help direct analysts to more effective changes and can save resources by helping to limit changes to those that improve process safety. Since the model that was developed was focused on the steps in the process of performing RAB and the potential for errors in the process and not on the medical judgment that goes into the process, the project team feels that DES can also be effectively used in other processes that use radioactive materials.

¹ This represents the opinion of Dana Kelly of Twinbrook Applied Sciences Inc. who was contracted as a PRA expert during the RAB project.

ACKNOWLEDGEMENTS

The authors thank the personnel who generously allowed us to observe the treatment processes at the facilities we visited and who spent time reviewing the RAB model. Their time, knowledge, and expertise are most appreciated. We would also like to thank James R. Callan, Ph.D., Richard T. Kelly, Ph.D., and David A. Kobus, Ph.D. of Pacific Science & Engineering Group for all their input during model verification efforts. Thanks, too, to Dana Kelly of Twinbrook Applied Sciences Inc. for his expert input to the Risk Assessment comparison section and his support throughout the preparation of this document. Thanks to Margo Toth for putting together the early format of the document. Thanks also to Luci Salvi of the Army Research Laboratory (ARL) Human Research & Engineering Directorate (HRED) for her technical review. A special thanks goes to Laurel Allender, Ph.D., also of ARL HRED, for her guidance and expertise.

ABBREVIATIONS

DES	Discrete Event Simulation
HDR	High Dose Rate
HRA	Human Reliability Analysis
LDR	Low Dose Rate
MA&D	Micro Analysis & Design, Inc.
NMSS	Office of Nuclear Materials Safety and Safeguards
NRC	Nuclear Regulatory Commission (U.S.)
PRA	Probabilistic Risk Assessment
PS&EG	Pacific Sciences and Engineering Group
PSF	Performance Shaping Factor
QA	Quality Assurance
RAB	Remote Afterloading Brachytherapy
SME	Subject Matter Expert
V&V	Verification and Validation

GLOSSARY

Applicator: A medical instrument placed in or adjacent to a tumor through which a radioactive source will travel during treatment delivery.

Brachytherapy: A cancer treatment process that uses radioactive materials to retard or destroy tumors with ionizing radiation.

Discrete Event Simulation: A technique for simulating and studying processes that can be described by a sequence of events that each have a distinct beginning and end. It is particularly well suited to modeling series of human tasks.

Dosage Deviation: An administered radiation dose that differs from the prescribed dose.

Dwell: The position or time (duration) that the radioactive source is located during treatment delivery. The source will be located at a specific dwell position for a specific dwell time.

Effect: In Micro Saint, an expression that executes as a result of a task during model execution. Tasks use beginning, and ending effects.

Error Dependency: A condition that can affect the probability of initiating or catching one or more errors.

Function: The top level of the RAB process description from NUREG/CR-6125. Functions are broken down into *tasks* and *steps*. They equate to the *nodes* of the top level *network* in the RAB model.

High Dose Rate: An RAB treatment type using a high activity source to deliver a dose in 5-10 minutes. Low Dose Rate uses a lower activity source to deliver a dose in 2-3 days.

Misadministration: For RAB, an administration of a radiation dose

1. involving the wrong patient, wrong radioisotope, or wrong treatment site,
2. involving a sealed source that is leaking,
3. when one or more sealed sources are not removed upon completion of the procedure, or
4. when the calculated administered dose differs from the prescribed dose by more than 20 percent of the prescribed dose.

Network: In Micro Saint, a sequential relationship of nodes that simulate a system, activity, or process displayed as the network diagram.

Node: In Micro Saint, a graphical representation of a *task* (rounded rectangle) or a *network* (rectangle) in the network diagram.

Path: In Micro Saint, a sequential connection between two *nodes*, so that one follows another under certain specified conditions.

Programmatic Error: An error that can affect multiple *fractions* of a single patient.

Recordable Event: For RAB, an administration of a radiation dose when the calculated dose differs from the prescribed dose by between 10 and 20 percent.

Release Condition: In Micro Saint, an expression that is evaluated to determine whether a node can execute. A node can execute only when the value of the Release Condition is nonzero or true.

Remote Afterloading Brachytherapy: A *Brachytherapy* treatment in which a remotely controlled device inserts and withdraws the radioactive sources from *applicators* that have been placed in a patient.

Snapshot: In Micro Saint, an option that records the values of specified variables at particular points during model execution.

Step: The lowest level of task decomposition of the RAB process description from NUREG/CR-6125. Steps are broken down from higher level *tasks*. They equate to *task nodes* in the RAB model

Subnetwork: In Micro Saint, a network that is inside another network. In a model, all networks are subnetworks except for the top network.

Systematic Error: An error that can affect multiple patients.

Task: The second level of task decomposition of the RAB process description from NUREG/CR-6125. Tasks are broken down from higher level *functions* and broken into lower level *steps*. They equate to *subnetworks* in the RAB model.

Task Description: In Micro Saint, each task node is defined by *task time*, *release conditions*, beginning and ending *effects*.

Task Time: In Micro Saint, the simulated time for a *task*. It is usually based on a mean time, standard deviation, and a distribution.

Treatment Fraction: In HDR RAB, partial treatments to patients separated by a few days. The entire prescribed treatment dose is often delivered in multiple fractions.

2. INTRODUCTION

This report describes a study conducted by Micro Analysis & Design, Inc. (MA&D) for the U.S. Nuclear Regulatory Commission (NRC). The study investigates Discrete Event Simulation (DES) as a risk analysis tool for medical procedures using radioactive by-products, and is part of the effort by the NRC to identify the risk analysis tools that are suited to assessing these procedures. This study demonstrates that DES can be used to evaluate the risks associated with Remote Afterloading Brachytherapy (RAB) by modeling the treatment process and identifying the human errors that contribute the most to the risk of a misadministration during RAB treatments. In addition, specific hardware and software failure scenarios were used in the model to demonstrate how DES can be used to assess the effects of these types of failures on the process.

2.1 Background

Remote Afterloading Brachytherapy (RAB) is a cancer treatment process that uses ionizing radiation from radioactive materials to retard or destroy tumors. RAB uses a computer-controlled device to remove the radioactive materials from a shielded container and move them through tubes into applicators that are placed in or adjacent to a tumor. The device controls the dwell positions and the dwell times inside the applicator. The device then returns the radioactive materials to the shielded container.

The NRC regulates certain aspects of RAB as part of its mission to protect public health and safety. One area of regulatory concern is misadministrations. Misadministrations include the following: administering a radiation dose to the wrong patient, administering a radiation dose to the wrong location of the correct patient, or when the intended and delivered doses differ by more than 20 percent. Misadministrations are often attributed to human error, and the consequences can be severe or deadly.

On November 21, 1992, a patient treated with a RAB device died after the brachytherapy source was erroneously left in an implanted applicator following treatment. In the past nine years, other patients that were treated with RAB devices have received radiation doses that differed from the prescribed dose, or doses that were inserted in the wrong location. All of these events involved human error.

In 1995, the NRC published a human factors evaluation of RAB to identify the human errors and factors that could lead to misadministrations. The evaluation discusses issues important to the safe use of RAB, including personnel, facilities, supporting equipment, software, procedures, and training. The study was specifically designed to identify the factors that contribute to errors in RAB systems, to evaluate the impact of those factors on the performance of functions and tasks essential to meet system goals, and to prioritize functions and tasks based on their safety significance. The results of this study are in the technical document, NUREG/CR-6125, Human Factors Evaluation of Remote Afterloading Brachytherapy Vols. 1-3, 1995.

The NRC is currently developing guidelines and methods for implementing a policy called "Risk-Informed Regulation." In implementing this policy, the NRC expects to employ risk assessment methods, most of which rely upon traditional applications of fault trees and event trees. In addition, the NRC is interested in using risk assessment methods to the extent that is supportable by the current state of the art in risk assessment. The purpose of this project was to demonstrate that DES is an alternative risk assessment method that might be more appropriate than the traditional HRA/PRA methods for certain applications.

2.2 Objective and Approach

Discrete Event Simulation is a well-established technique for studying processes that can be described by sequences of events that each have a distinct beginning and end. The two primary objectives of the project documented in this report were (1) to determine the feasibility of using Discrete Event Simulation (DES) modeling to assess the effects of human errors and hardware and software failures on process safety with RAB in particular and with medical processes that use radioactive materials in general, and (2) to compare DES with other risk assessment methods for evaluating medical processes that use radioactive materials.

The approach for the study was to develop a DES model for a typical high dose rate RAB treatment process. The model used a human factors evaluation of RAB (NUREG/CR-6125) as the basic task analysis for the development of the model. Human errors and human responses to hardware and software failures were included in the model.

This report describes how DES can be used for risk assessment, and describes the specific use of DES in building a model for RAB and using it to assess the risk of misadministrations in the process. The intent is to present DES modeling as an option to be considered for assessing the risk of a medical process. The report is organized into the following sections:

1. Introduction

2. Simulation Overview - An introduction to simulation and modeling

3. Discrete Event Simulation - A description of DES

4. Using Discrete Event Simulation To Assess Risk And Improve Safety - An explanation of how DES can be used for risk assessment

5. Micro Saint: Discrete Event Simulation Software - A description of the Micro Saint DES modeling software

6. Using Discrete Event Simulation To Evaluate The Use Of Radioactive Material In medical Treatment Processes - A methodology for modeling the risk in a medical process

7. Remote Afterloading Brachytherapy Discrete Event Simulation Project - A description of the RAB model developed in this study

8 Comparing Discrete event Simulation To Other Risk Assessment Methods - A comparison of DES with other risk assessment techniques

9 Results - Results, conclusions, and recommendations

This report is written so that readers can focus on the portions that are of greatest interest to them without having to read the entire document. For that reason, some essential material is presented in more than one section. Sections 2 & 3 are intended to provide the reader with a basic understanding of simulation in general and DES in specific. These sections will provide valuable background for the rest of the report to readers who are not familiar with the topic of simulation. Those who are already familiar with these subjects could skip these sections or refer to them as needed. Section 4 explains how DES can be used for risk assessment. The explanation is presented as a general discussion of each DES topic related to risk assessment. More detailed descriptions of how DES is applied are presented in Sections 6 & 7. Section 5 describes the simulation software tool, Micro Saint, that was used in this project to develop the RAB model. This section will be useful to readers who are interested in the detailed methodology described in Section 6 but are not familiar with the Micro Saint simulation tool, or it can be used to reference specific issues with the software. Section 6 describes the methodology for using DES to evaluate risk in a medical process that was developed

during the course of this project. The methodology includes many examples from the RAB model itself. This section can be useful to the reader who knows about DES and/or Micro Saint without having read the previous sections or while referencing the previous sections. Section 7 documents the RAB model that was developed in this study. It is a detailed description of how the Micro Saint software was used to create the model. This section can be skipped by readers not interested in the specific details of model construction. Section 8 presents a comparison of DES as a risk assessment tool with traditional risk assessment techniques. This section should be read by those interested in looking at different techniques. Section 9 presents the results, conclusions, and recommendations of this study.

Introduction

3.SIMULATION OVERVIEW

A simulation can be defined as an abstract representation or model of a system. Simulations are used to help gain a better understanding of the system. They are usually only representations of a part of a system, and the type of simulation that is created depends on the information you are trying to determine about the system under study. Simulations can be grouped into those that are *static* or *dynamic* with respect to time.

Static simulations are simulations that represent a system at a particular time. Examples of static simulations are computer spreadsheets, logic models, and mathematical models that do not represent change with respect to time.

Dynamic simulations represent a system as it changes over time. An example would be the activities in a hospital emergency room or a radiation therapy treatment facility over the course of a day. Common types of these simulations are man-in-the-loop simulators and other dynamic computer-based models.

- Man-in-the-loop simulators represent the human-machine interface for part of a system. Personnel can interact with the simulator in real time in much the same way they would with the real system. Simulators of this type are often used for personnel training and for testing how personnel will react to specific scenarios. Common examples include flight simulators and reactor control room simulators.
- Dynamic computer-based models are used to study the dynamics of a real-world system. The system can range from a facility to a process that you want to study. Computer models can either be *continuous* or *discrete* with respect to how they represent the passage of time within the simulation.

Continuous computer models are used when the aspects of the system change continuously with respect to time. They often rely on differential equations for the rate of change over time. Some examples of systems that require continuous models are wind over the wing of an airplane, the location of a vehicle in space over time, chemical processes, and weather models.

Discrete computer models are used for processes that can be described by events that each have a distinct beginning and end. Examples include human operation of systems, maintenance operations, manufacturing processes, and medical processes, which are the topic of this study.

6. Discrete Event Simulation

Discrete event simulations use a computer model to describe a process that can be expressed as a sequence of events, each with a distinct beginning and end. Events can be any part of the process, such as, scheduled activities, or tasks that represent the flow of the process. The tasks are displayed schematically on a diagram called the task network diagram, which is the basis of the model. The components of a DES are discussed in the following sections:

- 3.1 Task Network
- 3.2 Simulation Clock
- 3.3 Events(external to the task network)
- 3.4 Entities
- 3.5 Release Conditions
- 3.6 Task Effects on the System
- 3.7 Variables
- 3.8 Branching Logic
- 3.9 Model Execution
- 3.10 Simulation Output
- 3.11 Monte Carlo Techniques and Discrete Event Simulation

6.1 Task Network

The task network is a graphical representation of the process being modeled. Tasks are represented in a diagram that shows the order of task execution within the process. A task network diagram is made up of nodes connected by arrows. Individual tasks are represented by rounded rectangles. Rectangles represent subnetworks. The potential sequences in which the tasks are performed are indicated by the arrows between the nodes.

Figure 1 is an example of a task network that illustrates the method used to enter a medical treatment plan into a treatment unit. The P in the diamond-shaped node represents the type of decision that is used to determine which path is taken (see section 3.8 Branching Logic).

Figure 1. Example of a task network diagram

6.2 Simulation Clock

The simulation clock tracks the simulated time as the model executes. When events are used to advance the clock in variable time intervals, the simulation is referred to as event-driven. In these types of simulations, the clock advances by the amount of time required to complete the event. Examples of event-driven simulations are human operations of a system and manufacturing processes. Models developed in Micro Saint (section 5) are event-driven variable-interval simulations.

6.3 Events

Events are aspects of a simulation that are used to represent the process. A task is an event that has a beginning, an end, and an associated execution time. Events can trigger other events and can also change the status of the task execution in the model, such as using a resource. Events can be used to control the flow of the model and collect data within the model. In the network diagram in Figure 1, the first task or node acts as a decision event to determine which method of plan entry to use.

6.4 Entities

An entity is an object that travels through the task network. Entities can represent a physical object, such as, a part on an assembly line, or can represent a person, such as, a patient moving through the steps of a medical treatment. A task will be executed when two conditions are met; (1) an entity has reached the task, and (2) the release conditions for the task are met. Once the task has finished, the entity continues on to the next task in the network. In this report, an entity usually represents a patient. An entity in the RAB model that travels through a series of tasks in the network represents a patient that is having that series of tasks performed on him/her as part of the RAB treatment process.

6.5 Release Conditions

When an entity reaches a task, the release condition determines if the task can be initiated. Release conditions are often expressed in terms of the availability of a resource, such as, a tool, room, worker, or the completion of another task. For example, the task of moving a patient into a treatment room cannot occur until a treatment room is available and the previous task has been completed.

6.6 Task Effects on the System

Individual tasks can affect the overall activity, process, or system being modeled. These effects can also be described by how a task uses resources. For instance, if a task requires a resource, such as, a doctor or treatment room, these resources would become unavailable once the task started. However, when the task is completed, these resources would then become available. Another effect that a task can have is to initiate an error. In the real world, an error is committed during the

execution of the task. In a discrete event simulation, an error is represented as having been initiated by the task. Once a task is completed in which an error was made, the error can affect the process.

6.7 Variables

For the purposes of this document, the definition of variables refers to their use in computer codes with values being assigned to them. They can be used to track the status of the model and to record almost any value that is calculated during model execution. For example, variables can record how many patients are run through a model, or can record the task where an error occurred. Variables can be evaluated and changed to provide a way for the events in the model to interact with each other. A variable could also be used to monitor a resource, such as a doctor. At the beginning of a task that requires a doctor, the variable would be checked in the release condition to see if a doctor is available. If a doctor is available, the task would begin and the value of the doctor variable would be changed to show that the doctor is not currently available for any other tasks. At the end of the task, the doctor variable would again be changed to indicate that the doctor is now available for other tasks.

6.8 Branching Logic

A branch in the network diagram is used when more than one path can be followed. In the network diagram in Figure 1, a branch follows the first node to three other nodes that represent the three ways to enter the treatment plan. The path that is actually followed when the simulation runs is often based on either a probability or a pre-established rule. Branching logic may also be used to follow multiple paths simultaneously.

6.9 Model Execution

When the model execution is started, an entity begins at the first task node in the model. If the Release Condition for that task is evaluated as “true,” then the task executes. The effect(s) that the task has on the system are evaluated based on the expressions in the task description. The changes are expressed in variables that can be used in other tasks in the model. Once the task is completed, the entity proceeds to the next task in the network diagram. When more than one path is available, the branching logic is used to determine the path the entity will follow. In general, the entire network diagram is traversed by the entity, and the model is completed when the entity reaches the end of the last task in the network. Models can have conditions that send entities through the network until a specified simulation time, or until a pre-determined number have completed the simulation.

6.10 Simulation Output

The data output for a simulation are specific values of model variables recorded at a specific time during the execution of the model. The recorded data are used to answer questions about the system being modeled. The output is similar to the results of an experiment. Data output can include measures of system effectiveness, or can be used for system diagnostics. Some examples of useful output are as follows: resource utilization, process cycle time, cost, and errors initiated.

6.11 Monte Carlo Techniques and Discrete Event Simulation

A Monte Carlo simulation can be defined as any simulation using random variables. Discrete Event Simulations are Monte Carlo-based because they generate random numbers which are compared against probability distributions throughout the model. This process is used to determine a value for a specific instance from the distribution. In DES, this technique is often used to determine the task execution time for a particular instance of a task or to determine which path to follow in the case of a probabilistic decision point. Each iteration of these points within the model will produce a different value from within each distribution.

7.USING DISCRETE EVENT SIMULATION TO ASSESS RISK AND IMPROVE SAFETY

The results of this study show that a DES model can be used to help understand the risk and to improve the safety of the RAB medical treatment process. Based on previous experience with modeling and the successful development of the RAB model, DES can be used to model other medical treatments as well. The following sections discuss using DES models to assess the risk and improve the safety of medical treatment systems:

- 4.1 System Dynamics**
- 4.2 How Errors Propagate Through a System**
- 4.3 Characterizing Hardware and Software Failures**
- 4.4 Variation in System Processes**
- 4.5 Effects of Errors and System Failures on Safety**
- 4.6 Process Changes**
- 4.7 Facility-Specific Models**

7.1 System Dynamics

The task and error analyses that are the basis of a DES model are static descriptions of the process in the sense that the actual flow and interactions cannot easily be represented. The dynamic aspects of the system are depicted by building a model that shows variability within the process flow. Building the model includes constructing diagrams that show the interactions between the tasks and the errors (see Section 6.3). Events that occur in the process may change the way that subsequent events occur. The dynamic depiction of the system provides a greater level of understanding than what is derived from the static descriptions of the process.

7.2 How Errors Propagate Through a System

A thorough understanding of how errors can affect a treatment process is a valuable tool for reducing the level of risk in the process. Errors can have many different effects on a treatment process and the effects of an error can propagate through the system. Errors made at one point in the process can affect subsequent decisions. The flow of the process can be altered when steps are erroneously skipped. Errors can also change how subsequent errors occur. A DES model of a treatment process can be used to simulate the dynamics of error propagation and how the system is affected.

Individual errors are represented at the point in the process where they can be initiated. Detecting those errors is also simulated at each point where they can be detected. For example, during the task of placing an applicator into a patient, the error of inadequately securing the applicator might be made. Probabilities are used to determine whether the errors are actually initiated and, if they are initiated, whether they are caught for each execution of the model. The effects that errors have can be modeled as alternative paths in the process and as dependencies between the errors (see Section 6.3.4). For example, an alternate path could be taken if the error caused part of the process to be skipped, such as the omission of a quality assurance check. A dependency between errors is illustrated when an applicator that was inadequately secured affects the probability that the applicator might move while the patient is being transported.

One of the outputs from this type of DES model could be a list of errors that were initiated but not caught. In this way, an understanding of the most commonly occurring errors and the points in the process where they are initiated can be developed. The types of errors with the highest rate of occurrence and the points in the process where most errors are initiated can be viewed as some of the greatest areas of risk in the process. Results of the model simulation then become the focal point for efforts to improve the safety of the process.

7.3 Characterizing Hardware and Software Failures

As with human errors, an important aspect of process safety is to understand the effects of potential hardware and software failures. One method of characterizing these failures in a DES model is to use their frequency of occurrence. Reliability data for a piece of equipment are often expressed as a probability of on demand failure i.e., the expected number of failures in a specified period of time, usually one year. Potential equipment failures can be included in a DES model and demand failure probabilities could be used to initiate their occurrence during model execution. In this way, the risk associated with equipment failures can be analyzed along with the human errors.

Another method of evaluating the effects of equipment failures is to initiate a specific scenario within the model. Scenarios based on equipment failures can be developed from real-life occurrences or from subject matter experts (SMEs). Each scenario could be built into a treatment model and executed by the analyst. This allows the analyst to evaluate the effects of an equipment failure on the process without having to execute the model until a frequency-based failure occurs. Analyst-initiated failure scenarios is the method that was used to represent equipment failures in the RAB model.

7.4 Variation in System Processes

Another aspect of a system that can be represented by DES is the variability within the system itself. Variability within medical processes can include differences across treatments and patients, such as, how many applicators are used, how many fractional treatments are used to deliver a radiation dose, or the need for life support. Uncertainty analysis can also be performed using DES to address potential variability in the estimate of error likelihood and equipment failure frequencies.

7.5 Effects of Errors and System Failures on Safety

A DES model of a treatment process that includes human errors and equipment failures can also use the consequences of those errors to predict treatment outcomes. The consequences that each error and failure can have on the outcome of a treatment can be included in a DES model by translating the errors and failures into effects on the patients and staff.

Often predetermined definitions exist for what constitutes acceptable and unacceptable effects on patients and staff. Definitions of terms, such as, “misadministration” and “recordable event” may come from a regulatory organization or may be determined by the facility performing the treatment. In either case, the consequences of errors and failures can be based on these treatment outcome definitions. A model that includes a translation from resulting errors and failures to consequences can be used to predict treatment outcomes.

7.6 Process Changes

Changes in a process can be proposed for many different reasons. Some changes are designed to increase the efficiency of the process, while others improve the safety of the process. Such changes may be based on analyses of the efficiency and/or risk of the process and may come from sources, such as regulatory or administrative bodies. The cost of implementing such changes can be great and the effect that such changes may have on the process may not always be fully understood. A DES model can be used to assess the effect that proposed changes have on system safety.

The proposed changes can be made in the model prior to spending the time and money to change the actual process. The model can then be used to predict whether changes designed to improve safety have the required effect, and whether changes for purposes other than safety have any indirect effect on the safety of the system. Using a DES model this way can save considerable time and money by helping to limit the changes to those that will improve the level of safety in a process.

7.7 Facility-Specific Models

DES models can be developed for treatments at specific facilities and for each treatment of interest. These facility-specific models can be used to increase the safety of processes in any of the ways that were discussed in this section. In addition, such models have an added use that applies specifically to individual facilities. These models can be used to assess system vulnerability to incidents reported from other facilities.

Incidents that occur at one facility are often reported so that other facilities can understand the incident. The methods for disseminating such information can range from official reporting to word-of-mouth between facility personnel. However the information is received, a facility can compare the information regarding the incident with their model of the process. This comparison allows the facility to assess the level of susceptibility of their process to incidents that occurred at other facilities.

8. MICRO SAINT: DISCRETE EVENT SIMULATION SOFTWARE

Micro Saint is a simulation software package for building models that simulate real-life processes. In the previous section of this report, a description was presented showing how DES can be used to gain a better understanding of the dynamics of medical treatments that use radioactive material. In this section, the basic DES components that make up the Micro Saint software tool are described.

Models can be relatively simple or complex. A simple, functional model can be built by creating a network diagram and entering task timing information for each task in the network. More complex models can also be built that include dynamically changing variables, probabilistic and tactical branching logic, conditional task execution, and extensive data collection^{3/4} all of which can be specified by choosing menu commands or providing expressions for Micro Saint to execute under specific circumstances.

Whether the model is simple or complex, the process of executing the model and generating statistics and graphs from the collected data is mostly automatic. Micro Saint uses random numbers to generate task times and routing choices specific to the current execution. After running the model, statistics charts, scatter plots, line or step graphs, bar charts, and frequency distributions can be used to analyze the collected data. In addition, the results files can be opened by spreadsheet or statistical packages for further analysis.

The purpose of this section is to provide a brief introduction to the Micro Saint simulation software.

TI
m
co
m
M
Sf

s

8.

M
an
on
to

h-
g
c

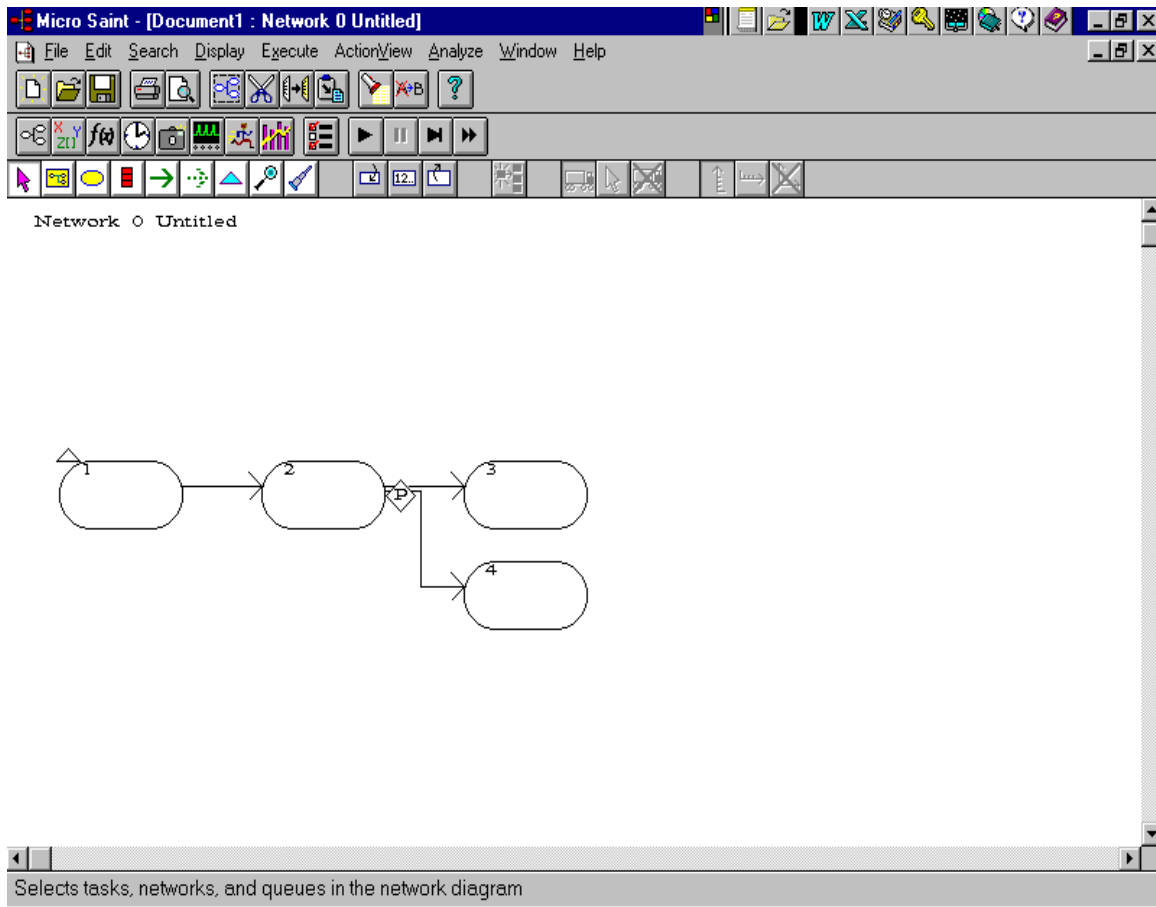


Figure 2. The Micro Saint user interface

Figure 2 shows the network diagram window of Micro Saint. It appears in a standard Windows format with pull down menus and button based tool bars. The window contains a sample network diagram of four nodes labeled 1 through 4 with a probabilistic decision node after node 2.

8.2 The Network Diagram

The process being modeled is represented in a diagram that depicts the tasks or activities that constitute the process. Each activity is represented by a graphical element, such as, a rectangle or rounded rectangle (also called nodes). The arrows between the nodes indicate the sequence in which the tasks are to be performed (Figure 3). The combination of nodes and arrows is called a network.

Figure 3. Schematic example of a task network

A rounded shape is called a task node. A rectangular shape is called a network node and indicates the presence of a lower level network sometimes called a subnetwork. Each subnetwork can contain either rectangular network nodes for other subnetworks or task nodes. Tasks are the lowest level of the network structure. In other sections of this report, the lowest level shapes will be referred to as task nodes to prevent confusion with other uses of the term ‘task’.

Task and network nodes are created in Micro Saint using the task and network tools. Users click on the network diagram with one of the tools to place a task (rounded rectangle) node or network (rectangular shape) node, and then continue clicking to place subsequent nodes as needed. The path tool is used to draw a path from each node to any other nodes that can follow it, and indicates the sequence of task execution.

Micro Saint also uses symbolic animation during execution. For example, when a particular node in the network has been reached, it is highlighted. The animation shows entities (items, people, etc.) as they move through the network. This type of animation is particularly useful in locating errors and in debugging a model.

8.3 Task Description

Tasks are the lowest level in a model network hierarchy and are described by specific parameters, such as, Timing Information (Section 5.3.1), Release Condition (Section 5.3.2), and Beginning and Ending Effects (Section 5.3.3) that relate the task to other system activities. An example of the Task Description dialog box is displayed in Figure 4. The description is for task number 1 (this number is internal to the Micro Saint software and does not affect or reference the process being modeled); a name for the task can be entered into the name field. Expressions for each of the task parameters can be entered in the labeled fields.

Figure 4. Task Description dialog box

8.3.1 Task Timing Information

Task timing information consists of the mean time for the task, the standard deviation, and a type of time distribution. In Figure 4, the task mean time is ten time units (hours, minutes, seconds, etc.), the standard deviation is one time unit, and the type of time distribution is Normal.

- The Mean Time is the average time required to complete a task. For example, if the task represents an activity, such as, “transfer patient to recovery room,” then the mean time to execute the task is the average time that it takes to transfer the patient. The mean time is used in conjunction with the time distribution to determine the simulated task execution time for each execution of the task.
- The Standard Deviation is used in conjunction with the time distribution, and controls the spread of a distribution.

- The Time Distribution defines the probability function used by Micro Saint to randomly generate execution times for a task. The mean time and standard deviation are used in conjunction with the probability distribution to determine the task execution time. In most cases, the execution time is not constant, but is variable within a range of values that can be represented by a probability distribution. Micro Saint supports more than 21 probability distribution types, including normal, rectangular, exponential, gamma, Weibull, Poisson, and others.

11.0.1 Release Conditions

Often, situations occur where a task cannot begin executing until certain conditions are met. A task can have resource requirements, such as, availability of a doctor or room, or other constraints, such as, time of day or availability of a part type that controls when the task can begin. In Micro Saint, the expression in the Release Condition field can prevent a task from executing until certain conditions in the model are met (e.g., the availability of a resource, the completion of another task). The Release Condition expression can be as simple as the value one (1) for tasks that always execute as soon as the previous task completes, or it may be a complicated expression in which several conditions are evaluated. Entities moving through the network cannot be released into a task for processing until the release conditions for the task are met.

11.0.2 Task Execution Effects

An execution effect defines how the task performance affects other aspects of the system. For example, the current state of the system may change when a task begins and then change again when the task ends. These changes are made using expressions in the Beginning and Ending Effects of a task description. In the example in Figure 4, the expression in the Beginning Effect of the task reduces the number of available doctors by one. The expression in Ending Effect increases the number of available doctors by one.

11.1 Controlling Process Logic

The basic order in which tasks are executed is defined by the arrows that are displayed between nodes. Alternatives are shown when more than one path originates from a single node. Task sequences can also be affected by conditions external to the network diagram. For example, a task can be started as a function of time. A diamond-shaped “decision node” automatically displays on the network diagram when more than one path follows a task. These decision points can be used to represent real world decisions or to control aspects of how the model works that may have little to do with the process being modeled.

The conditions that control the branching must be entered as expressions. Micro Saint provides the following decision types to ensure that real-world situations can be represented in the model:

- In a *Probabilistic* decision type, the next task to execute is determined by the relative probabilities of all tasks listed. Probabilistic decisions allow only one of the following tasks to execute.
- In a *Multiple* decision type, all of the tasks with conditions that evaluate to non-zero will execute. This allows for one or more tasks to begin execution based on rules that determine execution tasks.

- In a *Tactical* decision type, the next task to execute is the task with the condition that evaluates to the highest value. This allows for rule-based decisions. A Tactical decision type differs from the Multiple type in that only one following task is executed.

Variables and algebraic expressions can be used in the branching logic and the value of the variables can be changed by conditions in the model. This allows complete control and manipulation of the network flow. Figure 5 shows the Decision Description dialog box for decision 2. The decision type that is shown is Probabilistic, and the routing conditions for the two tasks that follow the decision, numbers three and four, have equal probabilities of execution (probability of Task 3 = .5, and probability of Task 4 = .5).

The screenshot shows a dialog box titled "Description of Decision" with a close button in the top right corner. Below the title bar is an "Edit" button. The main area contains the following elements:

- "Looking at Decision" followed by a text box containing the number "2" and navigation buttons "<" and ">" to its right.
- "Task Name" followed by an empty text box.
- "Decision Type" followed by a dropdown menu currently showing "Probabilistic".
- "Next Task:" followed by "Routing Condition:" and a button labeled "^ More ^".
- A list of tasks with their routing conditions:
 - Task 3: ".5;"
 - Task 4: ".5;"
 - Five empty rows for additional tasks.
- At the bottom, there are four buttons: "Accept", "Cancel" (with a red X icon), "Help" (with a question mark icon), and a button labeled "v More v".

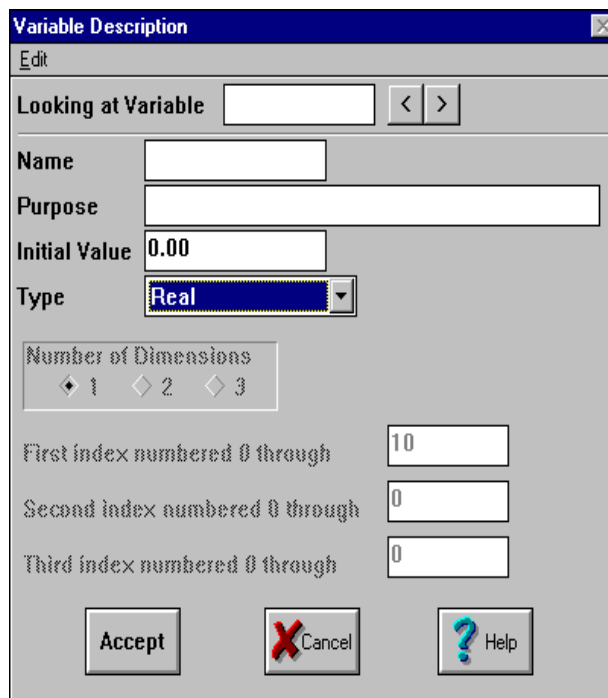
Figure 5. Decision Description dialog box

14.1 Variables

Variables are an important element of a model because they record many aspects of the system. You can define variables to record whatever is appropriate for the model¾how many patients were treated, the errors that occurred, whether a scenario is On or Off, etc. Variables give different tasks and scenario events a way to interact with each other.

Variables are created in the Variable Catalog by specifying the following information in the Variable Description dialog box (Figure 6):

- Name
- Purpose (optional)
- Initial value (if the variable is an array, this applies to all elements) (an array is a set of values)
- Type (integer, real, array of integers, array of real numbers)
- Dimensions, if the variable is an array (one, two, or three dimensions) and the size of each dimension



The image shows a 'Variable Description' dialog box with a blue title bar and a close button. It contains several input fields and controls:

- Looking at Variable:** A text field with empty space, followed by left and right arrow buttons.
- Name:** A text field with empty space.
- Purpose:** A text field with empty space.
- Initial Value:** A text field containing '0.00'.
- Type:** A dropdown menu with 'Real' selected.
- Number of Dimensions:** A group box containing three radio buttons labeled '1', '2', and '3'. The '1' radio button is selected.
- First index numbered 0 through:** A text field containing '10'.
- Second index numbered 0 through:** A text field containing '0'.
- Third index numbered 0 through:** A text field containing '0'.
- Buttons:** 'Accept', 'Cancel' (with a red X icon), and 'Help' (with a question mark icon).

Figure 6. Variable Description dialog box

19.1 Expressions

An expression can be a calculation, formula, function, or statement that supplies a value or performs an operation. Expressions can be used to supply numeric values, such as, mean times or true/false values, such as, those used in Release Conditions. They can also be used to make changes in the state of the model, such as, Beginning Effects and Ending Effects.

Each expression in Micro Saint must end with a semi-colon and can include any of the following elements:

- . Constants
- . Variables
- . Functions (groups of expression that can be called or referred to)
- . Comments
- . Mathematical operators (+, -, *, /, ...)
- . Assignment operator (:=)
- . Adjustment operators (+=, -=, *=, /=)
- . Logical operators (>, <, &, ==, ...)
- . If-then-else and while-do statements

28.1 Scenario Event

A scenario event is an event that is scheduled to occur at specific times (in simulation time) during model execution. These can be one-time events, or they can repeat at regular intervals. An example of a one-time event would be setting a variable at simulation time zero indicating the number of patients to run through a medical treatment model. Scenario events are also used to change variable values, thereby changing the state of the model.

Scenario events are defined by supplying the following information for each event in the Event Description dialog box (Figure 7):

- . Time of occurrence
- . Whether the event should repeat, and at what interval
- . Time at which you want the event to stop repeating, if applicable
- . The expressions you want executed at the specified time(s)

Event Description [X]

Edit

Looking at Event < >

Perform at Time

☐ Repeating
Repeat Interval

☒ Stop
Stop Time

Expressions:



Accept  Cancel  Help

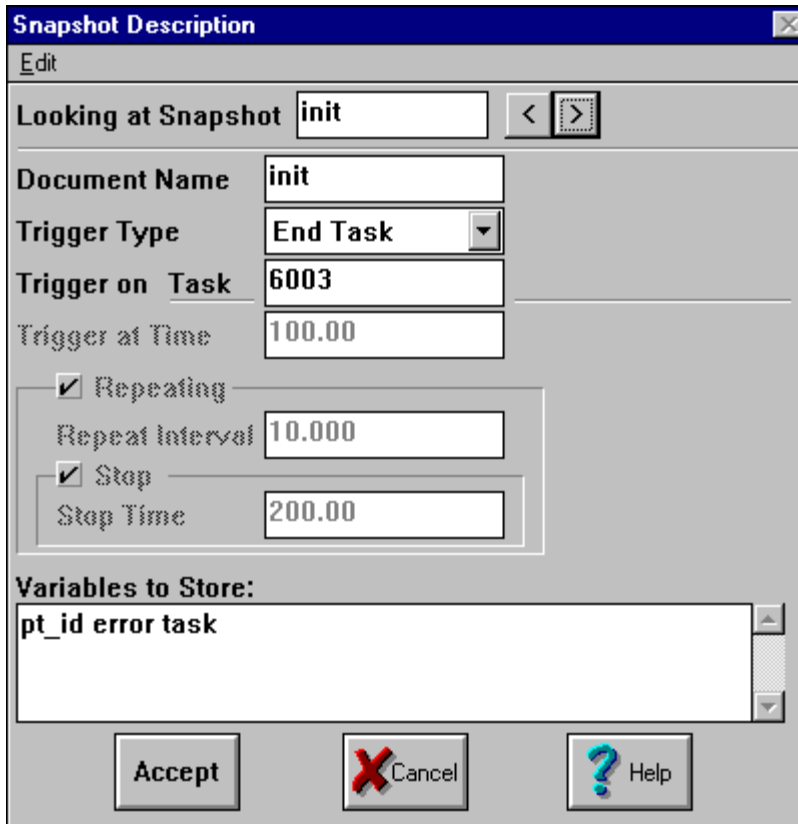
Figure 7. Event Description dialog box

32.1 Data Collection During Model Execution

Data are collected during the execution of a Micro Saint model using a feature called snapshots. Snapshots provide a way to collect values of variables at specified points during model execution. You can program snapshots to occur at specific clock times, when a task begins or ends, or when a model execution ends.

Snapshots are defined by providing the following information in the Snapshot Description dialog box (Figure 8):

- A name for the document where the data are stored
- The “trigger types” for the snapshot (End of Run, Clock, Begin Task, End Task)
- The number of the triggering task, if applicable
- The start time, stop time, and repeat interval, as applicable, if the snapshot has a clock trigger
- The names of the variables for which you want to record values



Snapshot Description

Edit

Looking at Snapshot < >

Document Name

Trigger Type

Trigger on Task

Trigger at Time

☒ Repeating

Repeat Interval

☒ Stop

Stop Time

Variables to Store:

Accept Cancel Help

Figure 8. Snapshot Description dialog box

In Figure 8 the example snapshot `init`, from the RAB model, traces all errors that are initiated during the model execution. The snapshot is triggered at the end of task 6003, and collects the values for variables `pt_id` (patient ID number), `error` (the number of the error initiated), and the task (the number of the task in which the error was initiated).

Once the snapshots have been defined, they can be set to On or Off during model execution. When they are turned On, the variable values are stored in a results file with the extension `.res`. After the file is opened, the Analyze menu in Micro Saint can be used to generate statistics and create graphs with the data. The data can also be imported into other statistical analysis packages.

38.USING DISCRETE EVENT SIMULATION TO EVALUATE THE USE OF RADIOACTIVE MATERIAL IN MEDICAL TREATMENT PROCESSES

This section presents a methodology for using DES to evaluate the safety of medical treatments that use radioactive material. Although the methodology was developed during the construction of the RAB model (Section 7), it is feasible to build models of other medical treatments. The methodology describes how to apply the concepts of DES using the Micro Saint modeling software to these types of medical processes.

In this section, a basic understanding of Micro Saint software is assumed and will be helpful in understanding the examples that are supplied. However, an understanding of the expressions that are given as examples is not required to comprehend the section's overall content. For more information on Micro Saint, see Section 5 or see the Micro Saint User's Manual.

38.1 Overview of the Methodology

In this methodology, the steps to build a basic DES model are described. These steps are based on the process followed during the construction of the RAB model. Each step of the methodology is presented in the order it was performed during the development of the RAB model. The order may be altered for different modeling efforts. Each step in the process is described in the following sections:

6.2 Collecting Initial Process Information

6.3 Building the Basic Network Diagram

6.4 Developing and Adding Error Distributions to the Model

6.5 Adding Error Dependencies

6.6 Adding Equipment Failures

6.7 Building Model Manipulation into the Event Queue

6.8 Collecting Model Execution Data

6.9 Verifying and Validating the Model

6.10 Recommended Additions to the Model

6.11 Using the Model for Risk Assessment

The remainder of Section 6 describes each of the steps in the methodology. Each step includes examples of how Micro Saint was used to model the RAB process.

38.2 Collecting Initial Process Information

The first step in developing a DES model is to understand the information requirements and how they can be collected. The first part of this section describes the information required to build the model (Section 6.2.1). The description of the information includes an explanation of how it is used in the model. The next section describes the sources that can be used to collect the required information (Section 6.2.2). The last part of this section provides an introduction to gathering information from subject matter experts (Section 6.2.3).

38.2.1 Information Requirements

Although a formal task analysis is not necessarily required, some method of identifying the individual tasks, task sequences, task times and distributions, and sequence logic in the process is needed. Additional information that is needed includes the human errors that could occur, the specific tasks where errors could be initiated, the tasks where errors could be caught, the mitigation procedures (correction of errors or reduction of negative effects), the probabilities of initiating and catching errors, and the error consequences.

Likewise, an understanding of the equipment involved is also necessary to model the role of hardware and software in a medical process. The equipment analysis should link the equipment to tasks from the task analysis, as well as, document, if possible, the reliability of equipment components, identify equipment failures that could affect the system, identify procedures for correcting equipment failures, and identify consequences of equipment failures.

What follows is a detailed description of the information that needs to be collected prior to the construction of a model. The order in which this material is presented does not necessarily imply an order for gathering the information. For efficiency, much of the information can be gathered at the same time.

Function/Task List

The treatment process that is to be modeled can often be decomposed into functions and tasks. A function is a piece of a process that is made up of more detailed parts organized into a logical grouping. Depending on the level of detail required for the analysis, the functions can be further broken down into subfunctions and tasks. A task is a discrete event or action that is part of the process. Tasks represent the lowest level of process decomposition.

The product of this step will be a function/task list that describes the process. The list will include a name or short title and a description for each function and task identified. The following example is a portion of the function/task list from the RAB model.

The RAB model is composed of the following five functions:

1. Patient Preparation
2. Treatment Planning
3. Treatment Delivery
4. Post-Treatment
5. Quality Assurance and Maintenance

The first function, *Patient Preparation*, is composed of the following five subfunctions:

1. Patient scheduling, identification, and tracking
2. Patient instruction
3. Life support monitoring
4. Applicator placement and stabilization
5. Patient transportation

The first subfunction *Patient scheduling, identification, and tracking*, of the first function is composed of the following tasks:

1. Receive a request for treatment of a specific patient
2. Schedule treatment for the patient
3. Notify the patient and staff of the treatment schedule
4. Arrange for transport of the patient to the treatment area
5. Track the transport and scheduling process
6. Receive and identify the patient when the patient arrives at the treatment area
7. Notify the staff that the patient has arrived
8. Arrange for identification, transport, and support for the patient within treatment area
9. Initiate and process any requests for subsequent scheduled treatments
10. Arrange for transport of the patient from the treatment area after treatment is completed

A complete list of functions, subfunctions, and tasks for the RAB model is provided in Appendix A.

Function/Task Sequence Logic

The sequence is the order in which the functions and tasks occur in the process. Many of the functions and tasks will proceed one after the other in a linear fashion. However, some may be performed in parallel and others may be repeated during the process. The function/task list and potential sequences are the two requirements for constructing a basic network diagram that represents only the fundamental steps in the process without human error or equipment failure considerations.

In cases where the process is not linear, the logic that determines the order of the tasks will need to be well understood. In the case of parallel tasks, you need to know at which point the path changes to two or more parallel paths, and whether one of the paths ends or where in the process the paths converge. For tasks or sequences of tasks that repeat, you need to know which tasks to repeat and frequency of the repetition. An example from RAB process is performing the same actions to place each of several applicators.

A path may also split in the process where only one route is followed. These alternate paths may occur when parts of the process are not performed for each treatment. Determining which path should be followed can depend on factors, such as, treatment variations, personnel judgment, etc. The information gathered in this step should include a detailed understanding of the logic, probabilities, and/or rules for these alternative sequences.

Task Times

In addition to the function/task sequence description, estimates of the time to execute each task is required to develop the model. Task times can be characterized in a variety of ways depending on the available information. In the ideal situation, sufficient information can be gathered to compute a mean, standard deviation, and time distribution for each task. In cases where this is not possible,

estimates of the time distribution and its parameters should be obtained from experts. See 6.2.3 for more information on how information can be collected from subject matter experts (SMEs).

Potential Human Errors and Initiation Points for Errors

In addition to the tasks, potential human errors that could occur during the process need to be clearly defined. Each task from the function/task list can be evaluated for potential errors that would adversely affect the process. During many actions, mistakes are often noticed immediately, and corrected almost without realizing it. Errors, such as these do not need to be modeled. Errors should also be evaluated for their effect on the flow of the process. Some errors affect the process by skipping certain steps. Other errors only affect the flow of the process, if they are caught.

For the RAB model, a list of all the relevant potential errors identified in the error analysis was created. Each error in the list included an identifying number and a short description. The following is a partial list of the errors from the RAB model. The complete list of errors from the RAB model is located in Appendix B.

Table 1. Partial list of errors from the RAB model

Error #	Description
6	Target locations in body misidentified.
7	Inappropriate applicator selected.
8	Applicator not placed correctly.
9	Applicator inadequately secured.
10	Connector incompletely mated to applicator.
11	Applicator position mislabeled.
12	Applicator distinction mislabeled (for multiple applications).
13	Applicator moved during transport.

Error Catch Points and Recovery Paths

The next step is to determine where in the process the errors could be caught; this is called a catch point. Each catch point should coincide with a task in the process. When an error is caught, it can be assumed that it will be successfully corrected or, if data is available, a probability of failing to successfully correct the error can be modeled. In the RAB model, it was assumed that any error that was caught would be successfully corrected. In either case, it is necessary to understand the actions that will be taken to correct the error because it may be possible to initiate or catch other errors while correcting an error. Often this involves returning to an earlier point in the process and redoing the tasks affected by the error. The analysis should include a link between each error and the task where it can be caught and the associated recovery path. In addition, it is necessary to understand whether or not the error can be caught before it causes an adverse consequence. It may not be necessary to model error catch points that occur after the error has already caused a serious problem.

During the process of identifying error catch points, it may be found that some errors cannot be caught at any point in the process. With respect to the safety of the process, this can be very valuable information. Any such errors should be carefully recorded along with a discussion of why each one cannot be caught. This is an example of identifying problems in a process through the building of a DES model.

Error Probabilities

Once it has been determined where an error can be initiated and where it can be caught, a method is needed to indicate when they are initiated or caught in the model. A probability can be tested against a random draw from a distribution to determine if an error will occur or be caught in the model. A zero probability means that the error never occurs and a value of one means that the error occurs every time.

Error probabilities can be built into the model in a couple of ways. A DES model will need values for these probabilities. If estimates of probabilities can be obtained for the errors within the process, then these values can be used as inputs to the model. A relative probability scale for the errors can also be developed. Such a scale would list the errors in order of relative probability of initiation, relative probability of being caught, and, if necessary, the probability of being successfully corrected. In order to use these scales within the model, relative values would have to be assigned to the errors.

Error Dependencies

Errors are not always independent of each other. The initiation of one error sometimes has an effect on the likelihood or probability of other errors. These relationships are called error dependencies. An example of an error dependency from the RAB process is when the error, *Applicator inadequately secured*, is initiated. This increases the probability of initiating the error, *Applicator moved during transport*. If the applicator is not sufficiently secured against movement, the chances are increased that it may move while the patient is being transported.

Another important type of error dependency are those errors that only apply to a specific scenario. An example from RAB is the error, "Applicator distinction mislabeled." This error can only occur if more than one applicator is being used. The information collected should include an understanding of the potential dependencies between the errors.

Potential Equipment Failures

Important pieces of equipment used in the process should be identified and associated with the steps where they are used. Included should be a description of what equipment failures can occur that can adversely affect the process. For the purposes of modeling, the correction of these failures can be tied to human errors. Human errors, such as, *Failure to notice equipment malfunction* and *Failure to take correct action following system malfunction* can be used to model the human responses to equipment failures.

One method of characterizing equipment failures in a DES model is to use their frequency of occurrence. Reliability data for a piece of equipment are often expressed in terms of a per-year failure frequency. If these data are available, potential equipment failures could be included in a DES model and failure frequency values could be used to initiate them during model execution.

Another method of evaluating the effects of equipment failures is to initiate a specific scenario within the model. Scenarios based on equipment failures can be developed from real-life occurrences or from subject matter experts (SMEs). Each scenario could be built into a treatment model and executed by the analyst rather than based on a failure frequency. This is the method that was used in the construction of the RAB model. See section 6.6 for more information on modeling equipment failures.

Error and Failure Consequences

The potential consequences of each human error and equipment failure can be useful for two reasons. First, they can be used as a check on the importance of each error and failure. If human error or equipment failure do not affect the process flow either by changing the path or stopping the process, or do not have a consequence that is of interest, then that error should not be included in the model. Second, consequences can also be used to translate errors and failures into process outcomes. This will allow the model to not only identify the errors or failures that have occurred, but also to predict a process outcome resulting from this information.

Level of Detail

The questions the model is designed to answer can affect the level of detail of the information that is gathered. If the model is intended to assess the process at a very general level, it is probably not necessary to model each keystroke in a data entry process. The required level of detail can also depend on whether a general model, suitable for multiple facilities, is to be created, or whether a site-specific model is desired. In the RAB model, the level of task detail was predetermined by the function and task list from NUREG/CR-6125.

The level of detail of the information that is gathered can also have significant influence on the usability of the model. It is important that the task and error descriptions are at the same level. If the tasks are stated generally and the errors are very detailed, there may be some problems mapping the errors to the tasks. For instance, the number of points in a process where a type of human error could occur may depend on the level of detail of the error description. An error with a general description, such as, *data entry error* can be initiated or caught at several places during the process. This can create a problem if it is important to know exactly which data entry error was initiated or caught within the model. Similarly, the description of an error can be so detailed that it is specific to a particular task.

10.0.1 Information Sources

This section describes the sources of the information that can be used to develop a DES model of a medical treatment. The description of each source includes the type of information that can be obtained. The sources presented include direct observation, written procedures, previous analyses, occurrence reports, regulations, equipment user guides, and subject matter experts. Any information collection effort will probably use combinations of many, if not all, of these sources.

Direct Observation

Much of the information can be collected by observing actual treatments. The tasks and task sequences in the process can be derived through observation. A record can be kept of the execution times for each task and can be used to develop mean times, standard deviations, and distributions.

Actual observation can also help to provide a better understanding of the dynamics within the process; these qualities cannot always be expressed in written material.

One limitation of direct observation is that the need for patient confidentiality may prevent the observation of the entire treatment process. The observation process may also be a very time consuming activity depending on how many treatments are viewed and how long each treatment takes. Also, it may not be possible to clarify issues during a treatment session, and people may behave differently if they know they are being observed.

Written Procedures

If written procedures exist for the treatment process, they can be extremely useful. The tasks and task sequences can be developed from the procedures without spending time observing actual treatments. Once the process is understood, then direct observations and subject matter experts will be useful in clarifying issues that may arise. Note, however, that the written procedures may not reflect how the task is actually being done, and the quality of the procedures may affect performance and error recovery.

Previous Task Analyses

If a task and/or error analysis has already been carried out for the process that is to be modeled, then reusing it may save considerable time and money. Any previous analysis should be carefully reviewed to determine what pieces of the required information may already exist and if the analysis is at the appropriate level of detail. Direct observations and discussions with SMEs can provide supplemental information that may be difficult to relate in the written task analysis.

Occurrence Reports

Occurrence reports describe events or incidents that have occurred. They can provide a general understanding of the types of human errors and equipment failures that can occur in the process and the consequences of those errors and failures. However, the reports may not be very detailed, and discussions with SMEs may be required to obtain a more thorough understanding of the event and the errors or failures. In addition, the reports typically present a very limited picture of the process, usually focused around the specific incident.

Regulations

Medical treatments involving the use of radioactive material are regulated. These regulations can often be used to define the risk analysis questions that are to be answered using the DES model. For instance, the regulations may define treatment outcomes that must be reported to the regulatory authority. The DES model can be used to assess the probabilities of these outcomes.

Subject Matter Experts

In the case of medical treatments involving the use of radioactive material, the personnel who can be considered experts in the process will usually be the radiation oncologists and the medical physicists. Much, if not all, of the required information can be obtained through discussions with the subject matter experts, or SMEs. SMEs can provide details and clarifications of task lists and task

sequences. SMEs can usually provide estimates of task times, but these estimates will probably not be as accurate as the empirical data that can be gathered by direct observation.

Perhaps the most important use of SMEs is for understanding the human errors. The personnel who provide the treatments will have a clear understanding of the potential errors, the points in the process where the errors can occur, and the points where the errors may be caught. The SMEs will also be able to describe the interactions or dependencies between the errors. SMEs may be able to provide estimates for the probabilities of errors as well as estimates of the potential consequences that each error may have. However, the ability and/or willingness to provide this type of information depends greatly on the individual expert. Some SMEs may be able to provide quantitative estimates of the probabilities while others may be willing to provide a ranking of error likelihoods. A basic understanding of the issues that arise while working with SMEs is recommended.

Equipment User Guides

The equipment used for medical treatments involving radioactive material often has a user manual or guide prepared by the manufacturer. This documentation can be used to create a list of the equipment and an understanding of where each piece of equipment is used in the overall process. However, this documentation will not include the reliability or failure data for the equipment. Such information may be available from the manufacturer or from specific facilities using the equipment.

10.0.2 Estimates from Experts

During the process of gathering information for the construction of the DES model, experts may be able to provide estimates for some of the required information. Specifically, experts may be used to estimate the probabilities of initiating and catching errors because little empirical data are usually available regarding the frequencies of errors in these types of medical processes. Techniques that prevent the introduction of bias should be used to obtain this type of information. The use of these methods is widely documented; one of the most useful documents is NUREG/CR-5424, Eliciting and Analyzing Expert Judgment. This document includes specific techniques and instructions for obtaining probability estimates from experts. The approach should be thoroughly understood before attempting to gather this type of information.

10.1 Building the Basic Network Diagram

Once the necessary information is obtained, the construction of the model can begin. The first step is to construct the network diagram as a graphical representation of the process flow. This section describes the steps required to build the basic network diagram. Although different medical processes have different modeling requirements, examples from the RAB model developed for this study are provided. Similar modeling techniques should be used for other studies as well.

The function-task hierarchy section describes how the functions and tasks can be graphically represented in a Micro Saint network diagram (Section 6.3.1). The data collection and processing multiple patients sections describe how these function nodes were used in the RAB network diagram (Section 6.3.2). The branching logic section describes how decisions can be modeled and provides examples of the expressions used to control the RAB model (Section 6.3.3). The section on characterizing error initiation and catch points describes how errors were incorporated in the RAB network diagram (Section 6.3.4).

In this section, the term *function* refers to two different concepts. The first is as a part of the hierarchy of functions and tasks. Here, the term refers to high-level process descriptions that coincide with the network nodes in the diagram. Later it is used to refer to programming functions.

10.1.1 Function - Task Hierarchy

Functions are the highest level in the hierarchy of the network diagram. Each function from the function/task list can be represented by a rectangular function node in the diagram. Each node is connected to another node with a path and arrow that indicates the order in which the functions are performed in the model. The example in Figure 9 shows two function nodes from the RAB model, *patient prep* and *treatment planning*, connected by a path showing that function two executes after function one.

Figure 9. The arrow between the two functions in the RAB model showing the order of execution

Task Description

Edit

Looking at Task Show

Task Number Name

Task Timing Information

Time Distribution

Mean Time: Standard Deviation:

Release Condition and Task Execution Effects

Release Condition: Beginning Effect:

Launch Effect: Ending Effect:

Figure 10. Task Description dialog box for the *Count Applicators* task

Within each function, networks of subfunctions and/or task nodes are built in the same way. Each task identified during the information collection effort is modeled as a task node in the network diagram. Nodes are connected with arrows that show the order in which they are executed. Each task node contains the name, task execution time, and time distribution associated with the task. Figure 10 shows a Task Description dialog box from the RAB model; the purpose of the task is to count the number of applicators. Although the time distribution field indicates *Normal*, a triangular statistical function is being used in the mean time field to determine the task time. This reflects the data that was available for the RAB model.

10.1.2 Collecting Model Execution Data and Processing Multiple Patients

In addition to showing the flow of the tasks, the network diagram can include additional nodes that are not part of the process being modeled. These nodes can be used for collecting data during the execution of the model and for processing multiple patients. Although both of these can be done in many ways, the method used for the RAB model is presented.

In the top-level network diagram of the RAB model, the function nodes *Collect Data* and *Next Patient* were added. The *Collect Data* function node was inserted as the second-to-last function node and the *Next Patient* function node was inserted as the last function node. A path was then drawn connecting the *Collect Data* function node to the *Next Patient* function node, and from the *Next Patient* function node back to the first function node (*patient prep*) of the network (Figure 11).

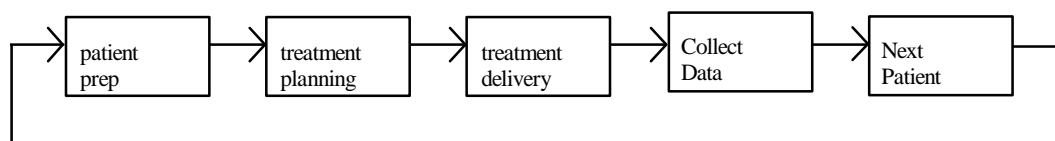


Figure 11. Addition of *Collect Data* and *Next Patient* functions

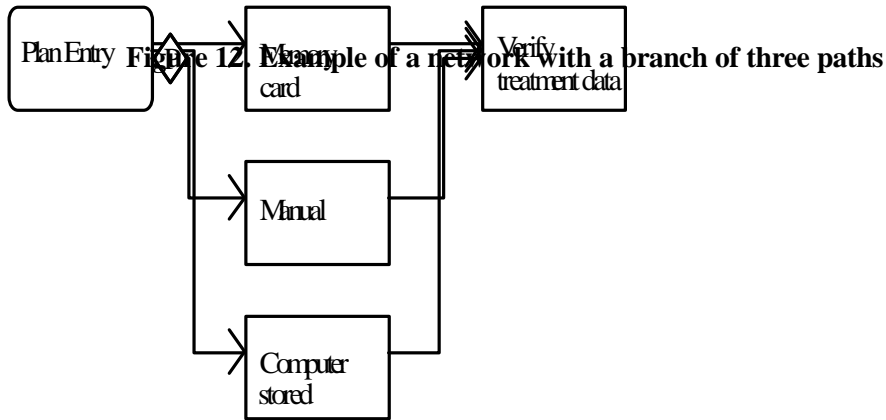
10.1.3 Branching Logic

When a process can follow more than one path after a specific function or task, branching logic is used by the model to determine which branch(es) will be followed. Multiple paths in the process are shown in the network diagram by more than one path line exiting from a single node. For additional information on decision types and routing conditions, refer to Section 5.4, or see the Micro Saint User's Manual.

The rest of this section shows examples of decision types that may be needed when building the network diagram. Also included is a description of how branching logic was used for multiple patient processing in the RAB model.

Example 1 - Probabilistic Decision

The first example from the RAB model is a network diagram with a branch of three different paths and a Probabilistic decision type. In this example, the treatment plan can be entered in the treatment system in three different ways: through the memory card, manually, or using a computer-stored plan (Figure 12).



Description of Decision

Edit

Looking at Decision: 3026

Task Name: Plan Entry Decision Type: Probabilistic

Next Task: Routing Condition: ^ More ^

3002	.5;	
.2a Mem		
3200	.25;	
.2b Manual		
3004	.25;	
.2c Comp		

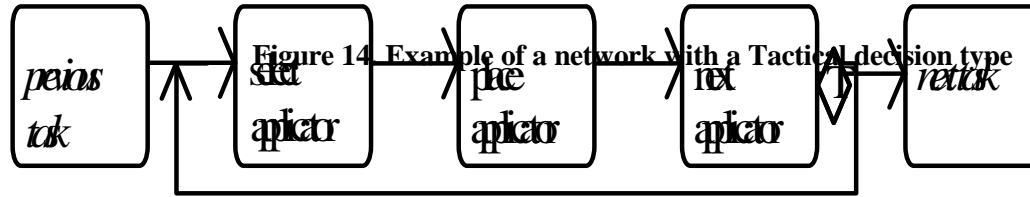
Accept Cancel Help v More v

Figure 13. Decision Description dialog box for a three-way branch

Figure 13 shows the Decision Description dialog box for the decision node. The decision type is Probabilistic and each routing condition expression is a value. In this example, the probability that the memory card will be used is 0.5, and the probabilities for the manual entry and computer-stored treatment plans are each 0.25. Micro Saint selects the path to be followed based on these probabilities.

Example 2 - Tactical Decision

The second example shows a sequence of tasks that are repeated. The network diagram in Figure 14 shows a task sequence for placing an applicator in the RAB process. The return path is used to



execute the same sequence of tasks for each applicator, if more than one applicator is used.

The decision type in this case is Tactical and the Routing Condition uses the variables `tot_appls` to record how many applicators are to be used and `appl_num` to record how many applicators were placed (Figure 15). The variable `tot_appls` is set at the beginning of the model execution to the total number of applicators needed for the treatment. In the *previous task* node, the variable `appl_num` is set to zero in the Ending Effect in the Task Description dialog box. This means that the number of applicators placed is set to zero.

In the Ending Effect of the *next applicator* node, the variable is incremented by one using the expression `appl_num += 1;`. The routing conditions check to see if all the applicators were placed by comparing the `appl_num` variable to the `tot_appls` variable. Figure 15 shows the Decision Description dialog box. If all applicators were placed, then the path to the *next task* node is followed. If less than the total number of applicators were placed, then the path to the *select applicator* node is followed.

Figure 15. Decision Description dialog box for Tactical decision

Example 3 - Tactical decision for multiple patients

As another illustration of a Tactical decision, the node called *Next Patient* is used to process more than one patient (Figure 16).

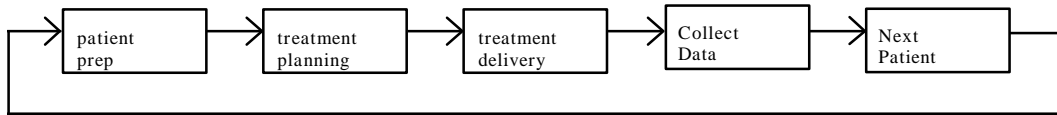


Figure 16. Task diagram for multiple patients

Within the *Next Patient* network node, two task nodes called *next patient?* and *no more patients* were created. The example in Figure 17 shows how these two nodes were connected. The circle with the number 1 represents returning to the beginning of the model to start the next patient.

Figure 17. Task diagram showing connection between nodes

A Tactical decision type was used to control the branching logic. The Description of Decision dialog box is shown in Figure 18. The variable *num_pts* is the total number of patients designated to run through the model and the variable *pt_id* counts how many patients have been run through the model. If all the patients are treated, then the model run ends. If the number of patients treated (*pt_id*) is less than the total number of patients (*num_pts*), then the path back to the beginning of the model is followed to begin the treatment for the next patient.

Figure 18. Decision Description dialog box showing routing conditions for multiple patients

In the model, two Micro Saint programming functions (see section 5.6) called **NEW_PT** and **INIT_PT_VARS** were used (capital letters were used to denote programming functions). The new patient programming function (Table 2) **NEW_PT** tracks how many patients have been through the model by incrementing the **pt_id** and calls the programming function **INIT_PT_VARS**, which resets the variables for counting errors. The call for the **NEW_PT** programming function can be put in the Beginning Effect of the very first node in the model so that it is executed for each new patient.

Table 2. New patient programming function code

NEW_PT function

```
pt_id+=1;
INIT_PT_VARS;
```

Matrices of variables called arrays² are used in the model to record the errors for each patient (see Section 6.3.4, Characterizing Error Initiation and Catch Points). These variables are reset to zero in the INIT_PT_VARS programming function (Table 3). For additional information on array variables and Micro Saint functions, refer to the Micro Saint User's Manual.

Table 3. Initialize Patient Variables programming function code

<p>INT_PT_VARS function</p> <pre> error:=1; while (error<total errors) do initiated[error]:=0, initask[error]:=0, caught[error]:=0, caughttask[error]:=0, error +=1; </pre>

10.1.4 Characterizing Error Initiation and Catch Points

This section describes how error initiation and error catch points can be created in a network diagram. The description includes single and multiple error initiations, single and multiple error catches, and combined error initiations and catches. In addition, examples of different error initiation paths and error recovery paths are presented. Section 6.4 describes the routing conditions and examples of expressions from the RAB model that can be used to control the error branching. The examples are very detailed and are included to provide a clear understanding of how this part of the methodology works.

Error Initiations

Error initiation can be modeled with a branch that uses a Multiple decision type and a node for representing the error. A Multiple decision type is used because more than one error may be initiated or caught at a single task. The probabilities of initiation and catch for each error are used within the task to determine when the specific errors are initiated and when they are caught. This is described in greater detail in section 6.4.

Using a consistent convention for naming the error nodes is suggested. In the RAB model, each error initiation node has the name +*HE* followed by the number of the error. For example, a node with the name +*HE1* was created to indicate a node where error number one can potentially be initiated. In the network diagram, task nodes where the error can occur were connected with the error initiation node. In Figure 19 the error +*HE1* can be initiated by task node 1. If the error does not occur, the path from task node 1 to task node 2 is directly followed. If the error does occur, the path to error node +*HE1* is followed. The path from the error node is then followed to the appropriate node elsewhere in the model. In the example, the error node is connected to task node 2.

² The term *array* is a programming term for a set of variables that has the same name but is indexed by number. For example, the array variable "patient[8]" indicates the eighth position in the patient array.

Figure 19. Path connection for the error initiation node

If more than one error can be initiated from a task, then a path from the task node to each of the error initiation nodes is used. Figure 20 shows that errors +*HE1* and +*HE2* can be initiated from task node 1.

Figure 20. Path connections for multiple error initiated from a single task

Error Catches

For each point where an error could be caught, the name *-HE* was used followed by the error number. A node with the name *-HE1* means that error 1 could be caught. Error catching (or mitigation) was modeled by creating a path from the error catch node back to the appropriate point in the process. Figure 21 shows an example of the network diagram for a catch point of *-HE1*. If the error is not caught, the path from task node 1 to task node 2 is followed. If the error is caught, the path from task node 1 to the error catch node is followed. The mitigation path returns to a logical point in the process for mitigating the error.

Figure 21. Network diagram showing an error catch

When more than one error could be caught in the same task, a node called *mitigation routing* and a Tactical decision type can be used to determine which error mitigation path should be followed. Figure 22 shows mitigation paths for each of two errors that could be caught, *-HE1* and *-HE2*. When more than one error is caught, the mitigation path that goes furthest back in the process is chosen.

Figure 22. Mitigation paths for two errors that could be caught

Multiple Initiations and Catches

Some processes may contain tasks in which errors can be both initiated and caught. These instances can be modeled with a combination of error initiation and catch nodes, a mitigation routing node, and an associated recovery path. Figure 23 shows an example in which error 4 can be initiated and error 2 can be caught by task node 1. The *mitigation routing* node is used to determine which path is followed based on which error was initiated or caught. If error 4 is initiated, the path to the next

node, task node 2, is followed. If error 2 is caught, even if error 4 is also initiated, then the error mitigation path is followed.

Figure 23. Example of a network diagram showing initiated and caught errors

10.2 Developing and Adding Error Distributions to the Model

This section describes the logic and expressions that control how errors can be initiated and caught within a model based on their probabilities. Examples of the expressions are included with the explanations of the modeling logic for those interested in the details of the methodology. However, understanding the expressions is not necessary to comprehend the overall content of this section.

The first section discusses the development of two probability distributions: one for probabilities of initiating errors and one for probabilities of catching errors (Section 6.4.1). The next section discusses how these distributions can be used within Micro Saint to determine whether errors are initiated or caught within a model. Two functions for that purpose are presented in Section 6.4.2. The last section continues with examples of the expressions that are used to control the branching at each point in the network where errors can be either initiated (Section 6.4.3) or caught (Section 6.4.4).

10.2.1 Probability of Error Initiation and Catch Scales

The information requirements for the error list described in section 6.2.1 include values for the probabilities of initiating and catching each error. The values for these probabilities could be unique for each error. In the RAB model, over 100 errors with their corresponding probabilities are in the error list (Appendix B). For ease of estimation, the data were clustered into two seven-point scales of relative probability: a scale for the error initiation and a scale for error catching. The range of each scale included the following relative terms: very very low, very low, low, medium, high, very high, and very very high. This scale provided sufficient variability within the range of values to reasonably

cluster the data so that each error initiation and catch value could be represented by one of the seven points in each scale.

All seven points of each scale do not necessarily have to be used. In the RAB model, only two of the points in the error initiation scale were needed to represent the data. A variable was assigned to each point of each scale. Variable names were selected to represent each value on each scale, e.g. ivlow for very low initiation value and the name cmed for medium catch value. Variables for all 14 values from the two scales were created this way. These variables allow a user to change the values within each of the relative probability scales for different types of model analysis.

10.2.2 Initiating and Catching Errors

The next step in creating an error distribution is to develop a method for identifying whether errors are initiated and caught based on the associated values from each of the seven-point probability scales. This can be done by drawing a random number and comparing it with the probability value for a specific error. For example, suppose an error has a low probability of initiation and the ilow value from the probability scale is .01. If a random number between zero and one is drawn and is lower than .01, the error is initiated. In addition, expressions in the model can be used to record which errors were initiated, which errors were caught, and the task number where this occurred.

In the RAB model, two Micro Saint programming functions were developed to compare error probability values to random numbers. The first is named ERRORINIT (Table 4). This programming function is used to determine whether a specific error will be initiated. A random number programming function, random(), that returns a number from a uniform distribution over the range of 0-1 is used. This value is assigned to the variable, testrand, and is compared with the probability of initiation for the error, pinitiate[error]. If testrand is less than or equal to pinitiate[error], then the error is initiated. The programming function then sets a flag for the error initiation, records the number of the task at which the error was initiated, and resets the error caught flags.

Table 4. Error Initiation programming function code

<p>Function ERRORINIT testrand:=0; while testrand == 0 do testrand:=random(); if testrand <= pinitiate[error] then initiated[error]:=1, initask[error]:=task, caught[error]:=0, caughttask[error]:=0;</p>
--

The other programming function, ERRORCAUGHT (Table 5) determines whether a specific error will be caught. It compares a random number against the probability of catching the error if it has already been initiated. If the error is caught, the programming function sets the error caught flag, records the task number where the error was caught, and resets the error initiation flag.

Table 5. Error Caught programming function code

```

Function ERRORCAUGHT.
while testrand==0 do testrand:=random();
if testrand <= pcaught[error] & initask[error] <> 0 then
  caught[error]:=1, caughttask[error]:=task,
  initiated[error]:=0, initask[error]:=0;

```

The next step is to create the expressions for each error initiation and catch point that will be used in the model to determine when the specific error is initiated or caught at the specific location. This can be accomplished in many different ways. The following examples show how this was done for the RAB model.

In the RAB model, the expressions for initiating and catching errors are in four places for each error node. Three of these are the Task Description dialog boxes for the task node where the error can be initiated, the node following the error initiation, and the one or more error nodes for that task. The fourth place is in the Decision Description dialog box for the path branch to the error node.

Each of the following examples in sections 6.4.3 and 6.4.4 include the expressions as they were used in the RAB model. Each example refers to either the Release Condition or Ending Effect for a specific task node.

10.2.3 Error Initiations

Figure 24 shows a diagram illustrating an initiation point for the single error +HE1. Table 6 provides the expressions controlling the error initiation.

- In the Ending Effect of Node 1, the expressions set the task and error numbers, set the probability of initiation of this error to low and call the ERRORINIT programming function. If the error is initiated, the initiation counter, *initiated[1]*, equals 1.
- In the Decision node following Node 1, the Routing Condition checks the initiation counter and follows the path to the next node if the error has not been initiated, *initiated[1] = 0*. If the error has been initiated, the path to the error initiation node is followed.
- If the error is initiated, the expressions in the Ending Effect of the error node (Node 3) reset the error initiation flag. This is done for instances where the specific error may be initiated in more than one place in the model. The initiation counter is reset so that another initiation of the error is possible.

Figure 24. Diagram showing initiation point for the single error

Table 6. Error Initiation expressions for a single error

Location	Field in dialog box	Expressions
Node 1	Ending Effect	task := 1; error := 1; pinitiate[error] := ilow; ERRORINIT;
Decision node following Node 1	Routing Conditions	Next Node 2: initiated[1] == 0; Next Node 3 initiated[1] == 1;
Node 3	Ending Effect	initiated[1] := 0;

Figure 25 shows an error initiation point for +HE1 and +HE2. Table 7 provides the expressions

controlling the error initiation.

- **The expressions in the Ending Effect of Node 1 set the task and error numbers, set the probability of initiation of error 1 to low and call the programming function ERRORINIT. The same expressions are set for error 2 with a probability of initiation of error 2 set to medium. The variable mult_error counts the number of errors that were initiated.**
- **In the Routing Conditions in the Decision node following Node 1, if neither of the errors is initiated then the path to the next node is followed. If either error is initiated, the path to that error initiation node is followed. If both errors are initiated, then both paths to the error initiation nodes are followed. As a result, the patient entity splits to traverse both of the error paths.**
- **In addition to the error initiation counters, the expressions in the Ending Effect of each error initiation node (Nodes 3 and 4) decrement the multiple error counter to show that the path to that error initiation node is completed.**
- **The Release Condition for Node 2 evaluates to true only if all multiple paths were completed. This condition recombines any split entities to ensure that they are not traversing the rest of the network diagram.**

Figure 25. Error Initiation points for multiple errors

Table 7. Error Initiation expressions for multiple errors

Location	Field in dialog box	Expressions
Node 1	Ending Effect	<code>task := 1;</code> <code>error := 1;</code> <code>pinitiate[error] := ilow;</code> <code>ERRORINIT;</code> <code>error := 2;</code> <code>pinitiate[error] := imed;</code> <code>ERRORINIT;</code> <code>mult_error := initiated[1] +</code> <code>initiated[2];</code>
Decision node following Node 1	Routing Conditions	<code>Next Node 2 mult_error == 0;</code> <code>Next Node 3 initiated[1] == 1;</code> <code>Next Node 4 initiated[2] == 1;</code>
Node 3: +HE1	Ending Effect:	<code>mult_error -= 1;</code> <code>initiated[1] := 0;</code>
Node 4: +HE2	Ending Effect	<code>mult_error -= 1;</code> <code>initiated[2] := 0;</code>
Node 2	Release Condition	<code>mult_error == 0;</code>

17.0.1 Error Catches

Figure 26 shows a catch point for the single error -HE1. Table 8 provides the expressions controlling the error catching.

- The expressions in the Ending Effect of Node 1 set the task and error numbers, set the catch probability of error 1 to high, and call the function ERRORCAUGHT.
- In the Routing Conditions in the Decision node following Node 1, the path to the next node is followed if the error is not caught. If the error is caught, the path to the error catch node is followed. The mitigation path is then followed from the error catch node.

Figure 26. Error Catch for single error

Table 8. Error Catch expressions for a single error

Location	Field in dialog box	Expressions
Node 1	Ending Effect	task := 1; error := 1; pcaught[error] := chigh;ERRORCAUGHT;
Decision node following Node 1	Routing Conditions	Next Node 2 caught[1] == 0; Next Node 3 caught[1] == 1;

Figure 27 shows the catch points for errors *-HE1* and *-HE2*. Table 9 provides the expressions controlling the error initiation.

- The expressions in the Ending Effect of Node 1 set the error and task numbers, set the catch probability for error 1 to high, and call the function ERRORCAUGHT. The same expressions are used to set the catch probability of error 2 to very high. The variable mult_caught counts the number of errors that were initiated.
- In the Routing Conditions in the Decision node following Node 1, the path to Node 2 is followed if no errors were caught. If either error was caught, then the path to the appropriate error catch node is followed. If both errors are caught, both paths to the error catch nodes are followed, creating two entities.
- The expression in the Ending Effect for each error catch node (*-HE1* and *-HE2*) decrements the multiple error caught variable to indicate that the path is complete.
- The Release Conditions for the mitigation routing node evaluates to zero only if all multiple paths are completed. The Routing Condition from the mitigation path node determines the single mitigation path to follow. If both errors were caught, the mitigation path that goes farthest back in the process is followed.

Figure 27. Error Catch for multiple errors

Table 9. Error Catch expressions for multiple errors

Location	Field in dialog box	Expressions
Node 1	Ending Effect	task := 1; error := 1; pcaught[error] := chigh; ERRORCAUGHT; error := 2; pcaught[error] := cvhigh; ERRORCAUGHT; mult_caught := caught[1] + caught[2];
Decision node following Node 1	Routing Conditions	Next Node 2 mult_caught == 0; Next Node 3 caught[1] == 1; Next Node 4 caught[2] == 1;
-HE1	Ending Effect	mult_caught -= 1;
-HE2	Ending Effect	mult_caught -= 1;
Mitigation routing	Release Condition	mult_caught == 0;

Figure 28 shows an error initiation point for +HE4 and an error catch point for -HE2. Table 10 provides the expressions controlling the error initiation and catch points.

- The expressions in the Ending Effect of Node 1 set the task and error numbers, set the probability of initiation of error 4 to low, and call the programming function ERRORINIT. For error 2, the catch probability is set to very high and the programming function ERRORCAUGHT is called. The variables mult_error and mult_caught count the errors initiated and the errors caught, respectively.
- In the Routing Condition, the path to Node 2 is followed if no errors were initiated or caught. If either error 4 is initiated or error 2 is caught, then the path to the appropriate error node is followed. If both occur, the paths to both errors nodes are followed creating multiple entities.
- The Ending Effects for each error node (+HE4 and -HE2) will decrement the multiple error counters to indicate that the paths were completed.
- The Release Condition for the mitigation routing node evaluates to true only if all multiple paths were completed. The Routing Conditions for the paths from the mitigation routing node determine whether to follow the path to Node 2 or the error mitigation path. In all cases, if error 2 is caught, the error mitigation path will be followed. If only error 4 is initiated, the path to Node 2 will be followed.

Figure 28. Diagram showing error initiation and error catch points

Table 10. Error initiation and catch point expressions

Location	Dialog box	Expressions
Node 1	Ending Effect	<code>task := 1;</code> <code>error := 4;</code> <code>pinitiate[error] := ilow;</code> <code>ERRORINIT;</code> <code>mult_error := initiated[4];</code> <code>error := 2;</code> <code>pcaught[error] := cvhigh;</code> <code>ERRORCAUGHT;</code> <code>mult_caught := caught[2];</code>
Decision node following Node 1	Routing conditions	<code>Next Node 2 mult_error + mult_caught == 0;</code> <code>Next Node 3 initiated[4] == 1;</code> <code>Next Node 4 caught[2] == 1;</code>
+HE4	Ending Effect	<code>mult_error -= 1;</code> <code>initiated[4] := 0;</code>
-HE2	Ending Effect	<code>mult_caught -= 1;</code>
mitigation routing	Release condition	<code>mult_error + mult_caught == 0;</code>

The expressions described in the examples were designed to allow any number of error initiations and/or error catches to occur at any point within the model. This allows the modeler to express any combination of error initiations and catches that can occur during a process.

27.1 Adding Error Dependencies

This section describes how error dependencies can be added to a model. The results section of this report discusses the effect that the use of error dependencies had on the RAB model. It was found that the limited number of dependencies modeled had very little effect on the results. A more thorough use of dependencies may have a different effect. See section 9.1 for more information.

Error dependencies affect the probabilities of certain errors. If a specific condition has been met in the model, the probabilities of initiation or catch are changed for a specific error based on dependency.

In the RAB model, expressions for the error dependencies were entered in the Ending Effect for the task nodes where errors could be initiated or caught. The following example (Table 11) shows the dependency between error 9 and error 13. If error 9, *Applicator inadequately secured*, has been initiated, then the probability of initiating error 13, *Applicator moved during transport*, is increased. In the expressions below, error 13 can be initiated at task 140. If error 9 has been initiated, then the probability of initiation for error 13 is medium; otherwise, if error 9 has not been initiated, the probability of initiation for error 13 is low.

Table 11. Ending Effect dependency code example

```
task := 140;
error := 13;
if initask[9]<>0
  then pinitiate[error]:=imed
  else pinitiate[error] := ilow;
ERRORINIT;
mult_error := initiated[13];
```

Another type of dependency that affects how errors are initiated are those that are scenario-dependent. Errors that are associated with recovery from an equipment failure, for example, can only occur if an equipment failure has occurred. Likewise, errors associated with mislabeling multiple applicators can only occur if more than one applicator is used.

27.2 Adding Equipment Failures

Including equipment failures within the model can be achieved in two different ways. One method of characterizing these failures in a DES model is to use their frequency of occurrence. Reliability data for a piece of equipment are often expressed as a probability of “on demand failure” i.e., the expected number of failures in a specified period of time, usually one year. Potential equipment failures can be included in a DES model using failure nodes similar to the error nodes. The demand failure probabilities could be used to initiate them during model execution. Equipment recoveries could be modeled in the same way as the error catches. In this way, the effects of the equipment failures on risk can be analyzed along with the human errors.

Another method of evaluating the effects of equipment failures is to initiate a specific scenario within the model. Scenarios based on equipment failures can be developed from real-life occurrences or

from subject matter experts (SMEs). Each scenario could be built into a treatment model and executed by the analyst. This allows the analyst to evaluate the effects of an equipment failure on the process without having to execute the model until a frequency-based failure occurs. The method of analyst executed equipment scenarios was used in the RAB model.

In the RAB model, a variable was set to either zero or one to indicate whether a specific equipment scenario was Off or On respectively. The variable was then used as part of the error dependency expressions for those errors associated with the equipment scenario. If the scenario variable was set to one, then the associated errors were initiated based on their probabilities. The scenario variable can also be used in a Routing Condition to select alternate paths that might be followed depending on the details of the equipment failure scenario.

27.3 Building Model Manipulation into the Event Queue

The Event Queue is used for scheduling events within the model. An event can be used to set variable values at specific times during the execution of the model (see Section 5 or the Micro Saint User's Manual). For the purposes of this type of model, the majority of the values are set at simulation clock time zero. The values are set at the beginning of the simulation to control the model execution.

Events of this type are defined in an the Event Description dialog box (Figure 29). The events are set to perform at time 0.00. There are default values for repeat intervals and stopping values, but since there is no check mark in the repeating box, these parameters are not activated. The values set by the event can be anything pertaining to the model. The example shows two values from the RAB model. The number of patients for this execution of the model is set to 1000 (num_Pts:=1000;), and the equipment failure scenario is turned off (eq_failure:=0;). Another such event in the RAB model is used to set the values for the error initiation scale and the error catch scale.

The image shows a screenshot of the 'Event Description' dialog box. The dialog has a title bar with 'Event Description' and a close button. Below the title bar is a menu bar with 'Edit'. The main area contains several fields and checkboxes:

- 'Looking at Event' field with the value '0.00' and navigation buttons '<' and '>'.
- 'Perform at Time' field with the value '0.00'.
- A group box containing:
 - 'Repeating' checkbox (unchecked) and 'Repeat Interval' field with the value '1.000'.
 - 'Stop' checkbox (checked) and 'Stop Time' field with the value '1000.00'.
- 'Expressions:' section with a text area containing:


```
num_Pts:=1000;
eq_failure := 0;
```
- At the bottom are three buttons: 'Accept', 'Cancel' (with a red X icon), and 'Help' (with a question mark icon).

Figure 29. Event Description dialog box

By setting these values, the Event Queue can be used to change the way the model is executed. The number of patients can be changed if model results for a specific number of patients is desired. The effects of equipment scenarios can be studied by comparing the results of different executions of the model with these scenarios turned On and Off. The effects of changes to the error initiation and catch scales can also be studied in the same way.

27.4 Collecting Model Execution Data

Data collection within the model captures details of model execution. These details take the form of values for specific variables at specific times during model execution. Recording the number of errors that were initiated at each task helps the analyst to identify tasks with a high frequency of error occurrence and tasks that are important to safety. For instance, it may be important to record each error that has been initiated and each error that has been caught. The node number at which an error was initiated or caught may also be important.

In a Micro Saint model, data collection uses a feature called snapshots. A snapshot records the values of variables at specific points during model execution. Each snapshot creates a results file containing the recorded data. For more information on snapshots and data collection, see Section 5 or see the Micro Saint User's Manual. In the remainder of this section, examples of how data were collected in the RAB model are presented. The data includes the information the user inputs into the model, the patient profile inputs, the errors initiated for each patient, and the errors caught for each patient. Several other types of data can be collected depending on the questions the model is to address.

In the network diagram for the RAB model, a rectangular network node was included called *Collect Data* (see 6.3.2). Within that network, four nodes were created called *Input Data*, *Pt Input Data*, *Collect Init Err*, and *Collect Caught*. Snapshots for these four nodes were used to record the general input into the model, the specific inputs for each patient, the errors that were initiated, and the errors that were caught respectively. The following example shows how the data collection is done for all the errors that were initiated.

Collect Error Initiation Data: an Example

A snapshot was created called *init* to collect the data for each initiated error. Figure 30 shows the Snapshot Description dialog box for this snapshot. As is shown in the Variables to Store field, the snapshot collects the values for each of three variables, *pt_id*, *error*, and *task*. The snapshot example is triggered on task 6003 and the Trigger Type is End Task. 6003 is the number of the node in the data collection network called *Collect Init Err*. The Repeating and Stop values are only needed when the Trigger at Time value is used. The expression *start(6003, tag)* was added to the "if" statement in the programming function *ERRORINIT* (the variable *tag* was not used by the RAB model but is required in the syntax of the expression by Micro Saint). This statement executes the *Collect Init Err* task each time an error is initiated in the *ERRORINIT* function. When the task is initiated, the snapshot records the patient identification number, the number of the error that was initiated, and the task in which it was initiated. These values are stored in a file called *init.res*. The file contains this information for all the patients that were simulated in the model. The model can record any

variable value in the same way.

The dialog box is titled "Snapshot Description". It features an "Edit" button in the top left corner. The main area contains several input fields and checkboxes:

- Looking at Snapshot:** A text box containing "init" and two navigation buttons (< and >).
- Document Name:** A text box containing "init".
- Trigger Type:** A dropdown menu set to "End Task".
- Trigger on Task:** A text box containing "6003" and a label "collect init err" to its right.
- Trigger at Time:** A text box containing "100.00".
- Repeating:** A checked checkbox with a "Repeat Interval" text box containing "10.000".
- Stop:** A checked checkbox with a "Stop Time" text box containing "200.00".
- Variables to Store:** A list box containing "pt_id error task".

At the bottom of the dialog are three buttons: "Accept", "Cancel" (with a red X icon), and "Help" (with a question mark icon).

Figure 30. Snapshot Description dialog box

Additional Model Execution Data Collection

The previous examples showed how data were collected for errors initiated for each patient. In the same manner, a snapshot was used to collect the error catches for each patient. The following are examples of other data collected in the RAB model. All of these data can be established from the two results files for error initiations and error catches. However, as the number of patients increases, the data analysis becomes more difficult. Organizing related data into smaller, more specific results files proved more useful.

- **Errors remaining.** The errors remaining are those errors that were initiated but not caught for each patient. In the RAB model, the array variable `inittask[error]` tracks the location at which each error was initiated. The values are reset when an error is caught or for each new patient. For each patient, the errors remaining and the tasks where they were initiated were collected from this array before it was reset for the new patient.
- **Total errors initiated.** A variable was used as a counter in the `ERRORINIT` programming function that records the total number of errors initiated.
- **Errors per patient.** The total errors initiated was divided by the number of patients that were simulated in the model to obtain the average number of errors initiated per patient.
- **Error initiations per task.** The error initiations per task were recorded to see which tasks have the most initiated errors. An array for all the tasks in the model was created and used in the `ERRORINIT` programming function to keep count of the error initiations for each task.

- **Initiations per error.** The number of initiations per error were recorded to determine which errors are being initiated most often. An array of the errors from the error list was used in the ERRORINIT programming function to count the number of times each error is initiated.

32.1 Verifying and Validating the Model

This section describes the process of verification and validation (V&V) for a DES model. Included are reasons why V&V is necessary, descriptions of the purpose, and a proposed schedule of when it can be performed during the modeling process. Section 6.9.1 discusses the issues of model verification and section 6.9.2 presents the issues of model validation.

32.1.1 Model Verification

The purpose of verification is to make certain that the model functions as it was intended. Verification is performed by testing and exercising the model throughout the model building process. The first way to test the model is to attempt to execute it. The Micro Saint software locates and presents any syntax errors that may have occurred while creating the model. Executing the model after any additions are made will ensure that syntax errors are addressed promptly.

Once the model executes without syntax errors, the next test is to trace the paths that the model takes to determine whether the decision nodes contain any logic errors. If the model is small, this process can be easy. However, as the size of the model increases, tracing each path can become time consuming.

Another way to test the model is to examine the results files to determine if the correct data has been recorded. If 100 patients were simulated, then data should be recorded for 100 patients. The files containing errors initiated and errors caught should also be reviewed. Some problems, such as, only one specific error ever being initiated, are obvious (many different errors should probably be occurring). The errors initiated and caught can be compared against the nodes where each occurred in the network diagram to determine if they were processed correctly. The error results files can also be examined to be certain that the errors being caught for each patient have actually been initiated for those patients.

The final way to test the model is to analyze the results files. This can be useful in two ways. First determine whether the results of the analysis seem reasonable; if not, the model may not be functioning as it was designed. Next, attempt to refine the data output; this may identify different kinds of data to collect or that the data needs to be presented in a different way.

32.1.2 Model Validation

The purpose of validating a DES model is to confirm that the model accurately represents the process that is being simulated. If empirical data on the process are available, the accuracy of the model can be assessed by comparing these data to the data output from the DES. However, such data are not always available. In most cases, the process of validation assesses the apparent validity or perceived accuracy of the model. SMEs, such as, radiation oncologists and medical physicists, are probably the best suited to perform this type of model validation. This is especially true if the model has been developed for a specific facility.

The basic validation process requires that SMEs review the model and identify any inaccuracies. Bias can easily be introduced into the questions that are asked of experts. For example, asking the

question, “Is this network correct?” can lead the SMEs to a biased response. Asking “Is this correct or not?” is a slightly better option, but still holds significant potential for bias. Reviewing the documentation on how to best use the judgment of the SMEs for validation may prove useful. Refer to NUREG/CR-5424, Eliciting and Analyzing Expert Judgment, for some helpful suggestions.

Validations can be done more than once during model construction. In fact, it can be extremely effective to schedule validations at various intervals throughout the process. Steps in model construction should be validated as soon as they are completed, if subsequent work is based on them.

A validation should occur as part of the task analysis. Although this step occurs before the model construction begins, it is included here with model validation as a necessary step in model development. Ensuring the accuracy of the data at this point can save considerable time and effort later during model construction. Even if an existing task analysis has been used, the validity of this information should still be assessed. The personnel who performed the task analysis may have used SMEs to validate the content as part of their process. Understanding their validation procedure may be useful. If a newly-constructed task analysis has been used, then the accuracy of the data collected should be determined.

A validation should also be performed when the network diagram has been constructed. This validation includes the function and task hierarchy, as well as the error initiation and catch points. Even though the accuracy of the data collected will be assessed prior to the construction of the network diagram, on paper these data represent a static description of the process. Building a dynamic computer model of the relationships between the tasks may require some interpretation of these descriptions. The SMEs involved in the RAB project were easily able to understand the network diagrams and, when asked to compare them to the process, were able to assess their accuracy.

The final validation uses the model execution data results. One method is to determine what parts of the process the SMEs believe contribute the most to the level of risk. This information can then be compared with the list of tasks where the most errors occur as predicted by the model. Discrepancies can be openly discussed with SMEs. However, this process can be very subjective since the answers given by the experts depend greatly on their perception of what constitutes risk and their experiences in specific facilities. For example, medical experts from a facility that performs patient prep, x-rays, and treatments, all in the same location, may not see the practicality of an error, such as “applicator movement caused by patient transport.”

32.2 Recommended Additions to the Model

The methodology has, thus far, described the construction of a model that includes the basic tools for assessing the safety in a medical treatment. This section describes several additions that can be made to this fundamental model based on the specifics of the medical process or the needs of the analyst. The additions include error consequences (Section 6.10.1), quality assurance and maintenance processes (Section 6.10.2), time-based frequencies (Section 6.10.3), fractional treatments (Section 6.10.4), the patient profile (Section 6.10.5), and systematic and programmatic errors (Section 6.10.6). The description of each addition includes a brief explanation of its usefulness and of the changes that must be made to the model to implement the addition.

32.2.1 Consequences

The consequences of human errors and equipment failures can be used within the model to predict treatment outcomes for each patient. For example, the model can be used to predict whether a

dosage deviation occurs for errors that have the potential to result in incorrect dosages,. A random number can be compared to a predetermined distribution of dosage deviations to predict the level of the dosage error.

In the RAB model, the errors that were initiated and not caught during the treatment were identified. A dosage deviation distribution was then used to translate each of the remaining errors into treatment outcomes. To capture these outcomes, additional data collection tasks and snapshots were added to the model.

32.2.2 Quality Assurance and Maintenance

Many of the procedures used in medical treatments are related to quality assurance (QA) and maintenance. A variety of quality checks, calibration procedures, and maintenance activities can be scheduled. These processes can be modeled in the same manner as the treatment process, including function and task sequences and errors. However, all of these activities may not be performed before each treatment. For example, a radiation alarm may be checked before every treatment, while the calibration of a source activity may be done on a monthly basis. If these types of QA activities are to be modeled and are of varying frequencies in execution, then these frequencies will have to be expressed within the model.

32.2.3 Time-Based Frequencies

Many processes, such as QA, are scheduled based on the passage of time. The example given in the previous section is the source calibration that may be done on a monthly basis. For such processes, the model must keep track of the passage of time. One method of expressing time-based frequencies is to establish the passage of time based on the number of patients treated per time period.

In the RAB model, a variable was used to express patients per week and was set in the Event Queue. The model recorded the number of patients and used the patients per week to determine when a week of patients had occurred. In this way, the model determined how much time passed based on the number of patients that were processed. The QA functions were grouped in the network diagram based on execution frequencies. A Tactical decision type was used and the Routing Conditions for each branch were based on the QA execution frequencies translated from the number of patients.

32.2.4 Fractional Treatments

Some of the medical processes involving radioactive byproducts deliver the prescribed radiation dose to the patient in multiple fractions. The delivery of a series of fractions can be spread out over the course of several days. The methodology, to this point, has described the modeling of one treatment per patient. If multiple fractions are to be simulated, then the model must be altered to allow for a pattern of multiple treatments per patient.

In the RAB model, the Patient Profile was used to determine how many fractions would be included in the treatment of a patient. The NEW_PT programming function was used to start a new patient only after all the fractions for the last patient were completed. The error counters for programmatic errors were also reset independently from the other human errors. For more information on programmatic errors, see Section 6.10.6. The data collection function was also altered to collect the fractional data at the end of each fraction and the patient data after all fractions were completed.

32.2.5 Patient Profile

The Patient Profile can be used to model differences in the treatment for each patient. If the process calls for fractional treatments to be delivered, then the number of fractions for each patient can be determined in the Patient Profile. In the case of RAB where multiple applicators are often used, the Patient Profile can contain data on the number of applicators for each patient.

The Patient Profile was created as a Micro Saint function that was called by the NEW_PT programming function. Random numbers were used to set each of the parameters for each patient. For example, when the treatment for each patient included between one and five fractions, a random draw between one and five was used to determine how many fractions were included in the current patient's treatment.

32.2.6 Systematic and Programmatic Errors

The majority of the errors associated with these types of medical treatments affect only a single treatment of a single patient. As such, the error counters in the model can be reset for each new patient. However, some errors can affect more than one patient. These are called systematic errors and are usually associated with the quality assurance and maintenance functions. Systematic errors are either equipment problems that have not been identified or are errors made during the calibration of some piece of equipment. Each error from the error list can be assessed to determine which are systematic. The expressions can be modified so that the error counter for each systematic error that is not caught during a treatment is not reset.

Some of the medical processes that use radioactive byproducts deliver the prescribed radiation dose to the patient in multiple fractions. These fractions are modeled as separate treatments of smaller doses. For treatments that are given in multiple fractions, another error category may be needed. Programmatic errors are those that can affect multiple fractions of a single patient. They are usually associated with errors made during the process of planning the treatment. For each programmatic error that is not caught during a fractional treatment of a single patient, the expression can be modified so the error counter is not reset. However, the error initiation counters for programmatic errors need to be reset for each new patient.

32.3 Using the Model for Risk Assessment and Improving Safety

This section presents several ways the model can be used to assess the risk and help improve the safety of the process being modeled.

- **Errors that cannot be caught within the system**
During the initial information collection effort and network building phase, errors may be identified that can be initiated in the process but cannot be caught. Errors that cannot be caught can be used to identify parts of the process where safety improvement efforts should focus.
- **Understanding system dynamics**
Building a DES model provides a detailed understanding of the dynamics in a medical process. This level of understanding may not be achieved while performing a task analysis because of its static nature. The dynamic nature of a simulation makes it a valuable tool for analyzing task flow and alternate execution paths.
- **Most frequently occurring errors and the tasks where the most errors occur**

The most frequently occurring errors and the tasks where the most errors occur drive much of the risk in the process. If the model is executed using a large number of patients (such as 1,000), the results should provide a clear indication of the errors that occur most often and the tasks that have the highest risk of errors.

The process can then be reviewed with SMEs to determine why these errors occur most often and why these tasks seem to have so much potential for errors. If the SMEs agree that these errors and tasks represent areas of risk within the process, then decisions can be made about what can be done to reduce that risk. However, if the SMEs do not agree that these errors and/or tasks increase risk, then refinements to the model may be needed.

- . Evaluating the effects of hardware and software failures**
A model that includes equipment failure scenarios can be used to assess how these failures affect the safety of the process. For each scenario, the model can be executed to determine if the treatment process can adequately respond to and mitigate the equipment problem. The frequency of errors and process outcome results can be used as a measure of how the process is affected by each equipment failure scenario.
- . Evaluating risk in the process**
The consequences of human errors and equipment failures can be used to determine how the patients involved in the treatment are affected. When the model is executed using a large number of patients the results will show the proportion of patients whose treatment outcomes were unacceptable or unsafe. This value can be used as a measure of the risk involved in the process.
- . Testing changes to the process**
The model can be used to assess how changes in the process affect the level of safety. This is an extremely valuable function of the model. If a change to the process is proposed, that change can be included in the model. The model can then be used to assess the effect that the change has on the level of risk. This verification process can result in considerable savings of time and money by preventing the costly implementation of changes that may not decrease, or may even increase, the level of risk in the process.

39.REMOTE AFTERLOADING BRACHYTHERAPY DISCRETE EVENT SIMULATION PROJECT

This section describes the Discrete Event Simulation (DES) modeling project that was used to create the RAB model, and includes the following sections:

7.1 Model Background

7.2 Sources of Information

7.3 The RAB Model

7.4 Data Output Measures

7.5 Verification and Validation

7.6 Using the Remote Afterloading Brachytherapy Model

39.1 Model Background

The first step in this project was to determine the medical process that would best demonstrate the feasibility of using DES. The NRC Office of Nuclear Regulatory Research sponsored research programs to examine human error in two different medical processes for the treatment of cancer using radiation therapy. These research projects focused on Teletherapy and Remote Afterloading Brachytherapy (RAB). In Teletherapy, the cancerous tissue is selectively destroyed by exposure to an external beam of ionizing radiation. In RAB, a radioactive source is temporarily inserted into an applicator that has been inserted into the body near the cancerous tumor.

Teletherapy and RAB processes were reviewed using the technical reports produced by the two NRC-sponsored research programs:

NUREG/CR-6277, Human Factors Evaluation of Teletherapy Vols. 1-5. 1995

NUREG/CR-6125, Human Factors Evaluation of Remote Afterloading Brachytherapy Vols. 1-3. 1995

The two systems differed with respect to the information that was available in the function and task analyses. The RAB research data were more useful in terms of what is needed to develop discrete event models of human errors. There were also some indications that RAB usage was increasing for cancer treatment while Teletherapy usage was declining. The conclusion was that the RAB process would be better suited for the purpose of demonstrating the feasibility of using DES for risk assessment.

Once RAB was chosen as the focus of the study, the project team also needed to choose between specific types of RAB treatments. RAB procedures include High Dose Rate (HDR) and Low Dose Rate (LDR) treatments. HDR treatments use a high activity radiation source to irradiate the target tissue in 5-10 minute treatment sessions. LDR therapy uses low levels of radiation for treatment sessions that typically last 2-3 days. The HDR treatment type was chosen because it represented a higher risk to patients and staff over shorter periods of time.

The human factors evaluation of RAB also examined the processes surrounding the use of three different radiation afterloading machines built by two different manufacturers. The team selected the processes associated with the MicroSelectron afterloader.

Within RAB, classes of treatments also differ slightly in process and in the probability of human error occurrence. Bronchial and gynecological cancer treatments were chosen as the scenarios for the modeling effort. Bronchial treatment was chosen because it was a common use of RAB. In addition, the research indicated that bronchial treatments had the highest incidence for movement of the applicators after they are placed, therefore having additional potential for misadministration. The treatment process for gynecological cancer using RAB is also a relatively common treatment type. Based on these factors, the team determined that the model could be generalized in such a fashion to adequately represent both the bronchial and gynecological RAB treatment processes.

39.2 Sources of Information

Three primary sources provided the information used to develop the RAB model. These were the RAB task and error analyses (Section 7.2.1), the SMEs from Pacific Sciences & Engineering Group (PS&EG) who prepared the task and error analyses (Section 7.2.2), and the information gathered from visits to sites where RAB treatments are performed (Section 7.2.3). Additionally, information was collected from occurrence reports of RAB incidents (Section 7.2.4) and from the MicroSelectron RAB User's Manual (Section 7.2.5).

39.2.1 RAB Task and Error Analyses

The majority of the information used to build the RAB model came from the technical report NUREG/CR-6125, Human Factors Evaluation of Remote Afterloading Brachytherapy, Vols. 1-3, 1995. The task analysis from the report included all the functions, subfunctions, and tasks of the RAB treatment process that were used to build the network diagram (see Appendix A for a complete list). The majority of the task sequencing information was also taken from the task analysis. The error analysis from the report was used to derive all the errors used in the model (see Appendix B). In addition, most of the error initiation and catch points, as well as the error recovery paths, were derived from the error analysis.

39.2.2 Pacific Sciences & Engineering Group

The task and error analysis document, NUREG/CR-6125, was authored by PS&EG personnel. They were considered to be experts on the contents of the report, and their expertise was used in two ways: for verifying and validating the model (see Section 7.5.1) and as a source of supplementary information to the task and error analyses.

Error probabilities. PS&EG provided a review of the project team's estimates for the relative probabilities of initiating and catching human errors. The entire error list was reviewed to locate the points in the process where errors could be initiated and caught. Consensus was then reached on the estimates for the initiation and catch probabilities. The estimates are more reflective of relative probabilities across the errors than estimates of absolute values for each probability. The list of all the errors and associated relative probability estimates are included in Appendix B.

Error dependencies. PS&EG also provided input for the error dependencies used in the model. A dependency exists between two errors when the occurrence of one error affects the probability of occurrence of the other error. An example is when error 4, *Applicator inadequately secured*, is initiated, then the probability of initiating error 13, *Applicator moved during transport*, is increased. Another type of error dependency provided by PS&EG refers to those errors that only apply to a specific scenario. An example is error 12, *Applicator distinction mislabeled*. This error can only occur if more than one applicator is being used. There was no attempt to develop a comprehensive

list of error dependencies for the RAB process. Those that were provided were used to show how dependencies can be modeled. Appendix B includes the list of error dependencies.

39.2.3 Site Visits

Site visits were an invaluable part of the information collection effort. In addition to specific data that were gathered, observations of actual RAB treatments and discussions with practitioners provided a firsthand understanding of the complexity and variability of the RAB process. The data gathered included the frequency with which specific quality assurance and maintenance procedures were performed, the scenario for the software failure, and task sequences specific to the site.

The Quality Assurance and Maintenance function as described in the RAB task analysis, includes several different procedures. The frequency with which these procedures are performed has a direct effect on the initiation and catch rates of the QA errors within the model. The number of patients affected by systematic errors is directly affected by the frequency of the QA checks and how many patients are treated between the checks. However, the RAB task analysis did not provide specific frequencies for the performance of all of these QA procedures. Since this information was not available from the task analysis, the project team chose to have the model emulate the QA & Maintenance schedule from a single site. A list of these procedures and their frequencies is included in Appendix B.

The DES model uses a specific software error scenario to demonstrate how software errors can be represented in a DES. The scenario incorrectly calculates dwell times, and was provided by personnel during a site visit. For more information, see Section 7.3.7.

The site visits also demonstrated the variability of processes across different sites. The task sequences in the RAB task analysis (NUREG/CR-6125) represent a combination of the processes from several sites. The modeled process, which was built from the task analysis, and the processes at the visited sites differed from each other in some respects. The requirements of this project stated that a model be built based on the RAB task analysis. Making site-specific changes was not required to demonstrate the feasibility of using DES, so changes were not made to the task sequences based on the site-specific process variations. Instead, this finding was used as a basis for concluding that a general model may not be appropriate for analyzing the process of a specific facility (see Section 9.1).

39.2.4 Occurrence Reports

During this effort, a specific scenario was developed to represent a hardware failure. The hardware failure scenario is based on a real incident in which the radioactive source became detached from the cable. The scenario was developed from incident descriptions detailed in occurrence reports. For information on the hardware scenario, see Section 7.3.7.

39.2.5 MicroSelectron High Dose Rate Equipment User's Manual

When the basic network diagram was being built from the task analysis (NUREG/CR-6125 Volume 2), it was found that the description for manually entering a treatment plan was difficult to follow. In addition, differences existed between that description and the process shown in RAB video tapes from the Nucletron Corp. To avoid any conflict between these two sources of information, the process as described in the MicroSelectron HDR machine operating manual from the Nucletron Corporation was used. Therefore, the process of manually entering a treatment plan is not based on the description from the task analysis. The differences turned out to be minor and had little or no effect

on the model.

39.3 The RAB Model

This section discusses the assumptions (Section 7.3.1), functions and tasks (Section 7.3.2), human errors (Section 7.3.3), error consequences (Section 7.3.4), branching (Section 7.3.5), and QA frequencies (Section 7.3.6) in the RAB model, in addition to hardware and software scenarios (Section 7.3.7) and model input (Section 7.3.8). The RAB model was designed using the Micro Saint simulation software. The appendices to this document are found in Volume 2 and contain the following detailed information about the RAB model:

Appendix A: Function and task list

Appendix B: Human errors, dosage deviation probability distribution, and error dependencies

Appendix C: Micro Saint network diagrams

Appendix D: Micro Saint variables

Appendix E: Micro Saint functions

Appendix F: Micro Saint events

Appendix G: Micro Saint snapshots

Appendix H: Micro Saint output files

Appendix I: User's Guide for running the RAB model

Appendix J: Micro Saint code for the RAB model

39.3.1 Basic Assumptions

The task analysis presented in NUREG/CR-6125 is a compilation of RAB processes from many different facilities. The analysis describes a complete set of individual processes that can occur but are not necessarily used at every facility. To reduce some of this complexity within the model, only two types of treatments were considered. Therefore, the functions, tasks, errors, and overall model flow are designed to reflect the processes required for bronchial and gynecological treatments only. In addition, the model uses a single type of RAB treatment machine; the Micro-Selectron High Dose Rate (HDR) treatment machine from Nucletron.

39.3.2 Functions and Tasks

NUREG/CR-6125 Volume 2 identified a function and task inventory for RAB treatment. Functions were divided into tasks that were then divided into steps. This hierarchy is represented in the Micro Saint model as networks, subnetworks, and tasks. Because the term 'task' as it is used in NUREG/CR-6125 and in Micro Saint has different meanings, there is some unavoidable confusion in any discussion of the function and task hierarchy. The following table is provided in an attempt to alleviate some of the confusion.

Table 12. NUREG/CR-6125 vs. Micro Saint terminology

NUREG/CR-6125 terminology	Micro Saint terminology
Function	Network
Task	Subnetwork
Step	Task Node

The function and task inventory from NUREG/CR-6125 is shown in the following table.

Table 13. Function and task inventory for RAB

Function	Task
Patient Preparation	Patient scheduling, id, and tracking
	Patient instruction
	Life support monitoring
	Applicator placement and stabilization
	Patient transportation
Treatment Planning	Simulation with dummy sources
	Target volume localization
	Radiation prescription
	Dwell position localization
	Dosimetry
	Treatment plan selection and approval
Treatment Delivery	Treatment set-up inside
	Treatment set-up outside
	Treatment plan entry
	Verify treatment data prior to treatment
	Treatment session monitoring
	Treatment session control
Post-Treatment	Source guide tube disconnection
	Applicator removal
	Patient transportation
	Treatment verification
	Record-keeping
Quality Assurance (QA) and Maintenance	Source exchange - Remove an old source
	Source exchange - Install a new source
	Source calibration
	Door interlock test
	Radiation warning light test
	Autoradiography of source position
	Interrupt and abort tests
	Source positioning and timing
	Check of source guide tube - applicator assemblies
	Equipment and software updates
	Troubleshooting

The networks and subnetworks in the RAB model were taken directly from this inventory. The tasks in the inventory are represented in the model as subnetworks in the networks which represent each function. Each subnetwork consists of steps that were also listed in the RAB task analysis (Appendix

B in NUREG/CR-6125 Volume 2). Each task node in the model is named and numbered based on the corresponding section in the RAB task analysis. For example, the task node labeled .3 *Monitor intercom* of the Treatment Session Monitoring subnetwork refers to step number 3, *Monitor intercom for patient comments* of the *Treatment Session Monitoring* task from Appendix B in NUREG/CR-6125 Volume 2. The complete list of the functions, tasks, and steps from the RAB task analysis that correspond to the network, subnetwork, and task node levels of the network diagram is included in Appendix A.

39.3.3 Human Errors

Many of the task nodes within the model are associated with human errors. This section describes the human errors used in the model from the analysis described in NUREG/CR-6125 Volume 1. Included are the modifications that were made to the error descriptions in the error analysis and a description of the method used to include them in the model. Also described are how probabilities were used for initiating and catching the errors and how dependencies between the errors were modeled.

Error Descriptions

Appendix A in NUREG/CR-6125 Volume 1 contains error tables that show the most likely human errors that could occur during the RAB process. Included in these tables is information about the step where each error could occur, immediate effects of the error, effects the error might have on other steps, as well as the potential danger to patients and staff. With the exception of some errors associated with the Quality Assurance (QA) and Maintenance function, the errors from the table are used in the model with little or no modification. The modifications consisted of wording changes recommended by subject matter experts.

QA errors changed to match task level. Table 13 includes the QA as described in the task analysis. While tasks describe specific checks and tests, some of the QA errors describe only generic failures. An example of a generic failure is the error, *Failure to perform QA procedure*. The errors were changed to more closely match the level of detail found in the tasks in order to adequately describe the events that could occur. Table 14 presents the original error descriptions from the error analysis in the left column and the modifications that were made for use in the model in the right column. The complete list of the errors used in the model is included in Appendix B.

Table 14. Quality Assurance error modifications

General Error Descriptions from the Error Analysis	Specific Error Descriptions for the Model
Failure to perform QA procedure	Failure to perform door interlock test
	Failure to perform radiation warning light test
	Failure to perform auto-radiography
	Failure to perform interrupt & abort test
	Failure to perform positioning & timing test
	Failure to perform source-guide tube check
	Failure to perform source calibration
	Failure to recognize problem during updates
Failure to recognize QA problem	Failure to recognize problem during interlock test
	Failure to recognize problem during radiation warning light test
	Failure to recognize problem during auto-radiography
	Failure to recognize problem during interrupt & abort test
	Failure to recognize problem during positioning & timing test
	Failure to recognize problem during source-guide tube check
Failure to record QA problem	Failure to record and communicate QA problem for interlock test
Failure to report QA problem (combined “record” and “report” errors)	Failure to record and communicate QA problem for radiation warning light test
	Failure to record and communicate QA problem for auto-radiography
	Failure to record and communicate QA problem for interrupt & abort test
	Failure to record and communicate QA problem for positioning & timing test
	Failure to record and communicate QA problem for source-guide tube check
QA certification error	QA certification error for interlock test
	QA certification error for radiation warning light test
	QA certification error for auto-radiography
	QA certification error for interrupt & abort test
	QA certification error for positioning & timing test
	QA certification error for source-guide tube check
Failure to record performance of QA test	Failure to record performance of interlock test
	Failure to record performance of radiation warning light

General Error Descriptions from the Error Analysis	Specific Error Descriptions for the Model
	test
	Failure to record performance of auto-radiography
	Failure to record performance of interrupt & abort test
	Failure to record performance of positioning & timing test
	Failure to record performance of source-guide tube check

Error Modeling

Using the information from Appendix A in NUREG/CR-6125 Volume 1, the errors were mapped to task nodes in the network diagram following the conventions described in Section 6.3.4. Each task node in the network that could initiate an error was followed by an error initiation node with a name, such as +HE21, indicating that human error 21 can be initiated at this point. A task node followed by the error catch node

-HE21 indicates a point at which human error 21 could be caught. Opportunities to catch and mitigate errors were based on information contained in the task analysis and provided by subject matter experts. Figure 31 shows a portion of the RAB network diagram displaying potential error initiation and error catch points for errors 40, 42, and 43. The diagram represents the *Session monitoring* task from the function/task list. It includes four steps in the task nodes *Monitor console*, *Monitor camera*, *Monitor intercom*, and *Instruct*. It also shows the error initiation and mitigation paths for each of the error nodes. The numbers at the top of each node (e.g., 3075) are used internally by the simulation software. A complete set of network diagrams is included in Appendix C.

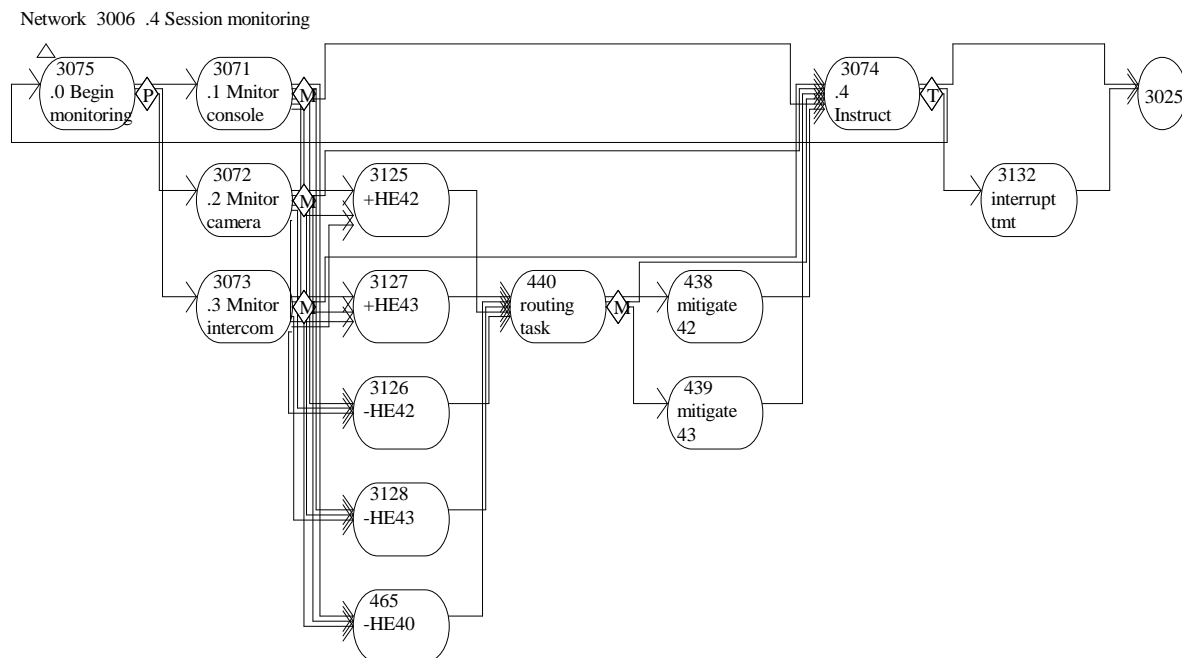


Figure 31. Error initiation and catch points

Error Probabilities

The estimated relative probabilities for error initiation and catch were measured on two seven-point scales. The scales included the following values: very very low (vvl), very low (vl), low (l), medium (m), high (h), very high (vh), and very very high (vvh). One initiation scale value and one catch scale value were assigned to each error based on the relative estimate for that error. The error probabilities and the scaled values for each error are in Appendix B.

A second set of SMEs were expected to verify the original information. These SMEs included practitioners who are considered by the NRC to be experts in the field of RAB. However, during the site visit with this group, the practitioners were unwilling to provide estimates for the error probabilities stating that any such estimates would be extremely inaccurate because of their own biases and the complexity of the RAB process.

Error Dependencies

Many of the events that could occur during the performance of a RAB treatment include errors that can increase or decrease the likelihood of other errors. These relationships are referred to as error dependencies. SMEs identified dependencies between many errors for which the occurrence of one error affected the probability of occurrence of one or more others. An example of this type of dependency is when error 4, *Applicator inadequately secured*, is initiated; then, the probability of initiating error 13, *Applicator moved during transport*, is increased. These dependencies are modeled by dynamically adjusting the probabilities of initiating and catching errors within the simulation.

Another type of error dependency represented in the model refers to those errors that apply only to a specific type of scenario. An example is error 12, *Applicator distinction mislabeled*. This error can only occur if more than one applicator is being used. Appendix B describes all the dependencies represented in the model.

No attempt was made in this project to obtain a comprehensive list of the dependencies between all the errors in the model. Those that are included in the model are for the purpose of demonstrating how error dependencies can be considered.

Programmatic and Systematic Errors

The majority of the errors on the error list can affect only a single fraction of a single patient's treatment (the RAB process often delivers the prescribed radiation dose to the patient in multiple fractions). However, two categories of errors can affect more than one treatment, if they are not caught. A systematic error can affect multiple patients before the error is caught and mitigated. A programmatic error could affect multiple fractions of a treatment for a single patient.

Each error in the model was evaluated to determine its category. The programmatic errors are mostly associated with treatment planning. If the treatment plan is incorrect, then all the fractions for that patient's treatment could be affected. The systematic errors are all associated with the Quality Assurance and Maintenance functions. Problems with the functioning of the RAB system could affect multiple patients. The error list in Appendix B includes a column indicating whether the error is considered systematic or programmatic.

Error Control

When an error is initiated, a variable that is used as a “flag” is set to a value of one (1) to indicate that the error was initiated. The model records all errors that were initiated and the task node where each error was initiated. When an error is caught, the initiation flag is reset to a value of zero (0) to indicate that the error can no longer affect the process. For the purposes of the model, the assumption was made that when an error was caught, it was also successfully corrected. At the end of the treatment for each patient, errors are checked to see which errors were initiated and not caught. These errors are then mapped to potential misadministrations (see Section 7.3.4).

Systematic and programmatic errors. The initiation flags for all non-systematic and non-programmatic errors are reset after each treatment fraction because these errors can only affect the treatment fraction in which they were initiated. The flags for the programmatic errors stay set for subsequent fractions of the same patient and will only be reset if the error is caught or if all the patient’s fractions are completed. The flags for the systematic errors remain set for all treatments of subsequent patients until they are caught.

QA and maintenance errors. QA procedures are designed to look for and catch problems. If there is a problem with some aspect of the system that a QA procedure does not catch, every patient that receives treatment until the error is caught can be affected by the problem. For example, one QA procedure is to perform a source calibration. If this procedure is omitted, all patients who are treated before the next calibration may be affected by a source strength error. This is represented in the model by setting the error initiation flag when the error occurs and resetting it only when the next appropriate QA task is performed.

39.3.4 Error Consequences

The consequences of errors in the model and the error analysis in NUREG/CR-6125 are based on the NRC definitions of medical misadministrations (Code of Federal Regulations). In general, a misadministration for RAB means that the radiation dose was not delivered as intended in the treatment plan. Three types of misadministrations are of interest for the RAB modeling effort: a radiation dose delivered to the wrong patient, a radiation dose delivered to the wrong treatment site, and a dose delivered that deviates from the level intended in the treatment plan. A dosage deviation is considered a misadministration when the dose delivered to the patient is more than 20% above or below the planned dosage. Dosage deviations are considered “recordable” events when the dose delivered is 10%-20% above or below the planned dosage.

Three types of misadministrations and recordable events are represented in the model: dosage deviations, wrong patient, and wrong treatment site. The SMEs who developed the task analysis were able to group the errors into categories based on the three different types of misadministrations. In the RAB model, errors that can cause the wrong patient and wrong treatment site misadministrations are termed “non-dosage” errors. All others were considered dosage errors and were divided into seven groups. These groups, called error categories, correspond to different potential levels of dosage deviation (Table 15).

Table 15. Error categories

Error Category Number	Error Category
0	Uncertain
1	No Dosage Deviation
2	Minimal Dosage Deviation
3	Moderate Dosage Deviation
4	Strong Dosage Deviation
5	Maximal Dosage Deviation
6	Inadvertent Exposure to RAB Staff

Error categories 2-5 constitute different levels of potential dosage deviations to patients. Error category 6 includes the errors that can cause inadvertent exposures to the RAB staff. Appendix B includes a column that lists the category assigned for each error.

The probability distributions in percentages for dosage deviation for each category are shown in Table 16. The table includes error categories 1-6 and the associated distributions. Each distribution is based on five different levels of dosage deviation from the planned treatment outcome: misadministration (>20% over or under), recordable incident (10-20% over or under), and normal outcome (within 10%). These are listed under the “Category Description” heading. As an example, error category 3 “Moderate Deviation” indicates that 12% of the time an error from this category will result in a dosage deviation within 10% of the planned dose. 20% of the time an error in this category will cause a dosage deviation from 10-20% over or under the planned dosage. 24% of the time an error in this category will cause a dosage deviation of more than 20% over or under that planned dosage. Each of the error categories listed in the table should be interpreted in this same manner. The numbers in the “Output Code” column are used in the results files from the model to indicate the treatment outcome.

Table 16. Probability distributions of dosage deviation

	Error Category	1	2	3	4	5	6
Output Code	Category Description	No Deviation	Minimal Deviation	Moderate Deviation	Strong Deviation	Maximal Deviation	Inadvertent Exposure
-2	> 20% Under	0	0	24	30	48	0
-1	10-20% Under	0	4	20	16	2	0
0	Within 10%	100	92	12	8	0	0
1	10-20 % Over	0	4	20	16	2	20
2	> 20% Over	0	0	24	30	48	80

At the end of each treatment fraction, errors with initiation flags that are still set are those that have not been caught. Each of these errors has the potential to cause a misadministration. Non-dosage errors that can cause either *wrong patient* or *wrong treatment site* misadministrations are reported in the model output. For errors that can cause dosage deviations, a random value from the appropriate probability distribution (Table 16) is used to determine the level of dosage deviation. The SMEs were not able to assign categories to a few errors. The model reports the consequences of these errors as uncertain.

Unlike most of the errors, some of those associated with QA cannot have consequences by themselves. The QA errors that include *failure to recognize a problem* and *failure to record and communicate a problem* can only occur if a problem exists with the system. In the model, these errors are dependent on the software and hardware scenarios and cannot be initiated unless such a problem exists. The errors associated with the QA checks included in the model (Table 14) cannot have any consequences unless the system shows a problem. Failing to perform a QA test does not have any adverse effects unless the QA test could have caught a system problem. In the model, the error types for these QA errors are dependent upon the hardware and software scenarios and are reported as not having a consequence unless a system problem exists.

39.3.5 Branching

In the previous sections, the manner in which the RAB task analysis information was used in the model was described. This section describes the branching logic that controls much of the flow of the process within the model. This includes error initiation and recovery paths, the frequency of the QA tasks, parallel sequences, task sequence control, selective task execution, and repeated tasks.

Error Initiation Paths

When an error occurs in a process and is undetected, two things can occur with respect to the flow of the process following the error. First, when a task is performed erroneously, the flow may not be affected by the error and will continue. Second, when a task is omitted, part of the process may be skipped due to the error. The RAB model represents both types of process flows. An example of the first process flow is if error 8, *Applicator not placed correctly*, is initiated, the process continues as if

no error has occurred. The error may cause a problem with the dosage delivered to the patient but does not interrupt the flow of the process. An example of the second process flow is if error 71, *Failure to perform door interlock test*, is initiated, the tasks that make up the door interlock test are skipped.

Error Recovery Paths

Although the RAB task analysis contains information about points in the process where specific errors could be caught, it does not describe the process of mitigating an error (i.e., recovering from or correcting the error). However, the process of mitigating an error that has just been caught usually requires repeating part of the process. Therefore, when an error is caught that requires redoing part of the process, the model uses a branch back to the appropriate point in the model to represent error mitigation. This is called an error recovery path. These are important because they create additional opportunities for errors to be initiated and/or caught as parts of the process are repeated.

Catching Multiple Errors

Within the model, more than one error can often be caught at the same point. Only a single recovery path was followed in these cases. When more than one error is caught at the same time, the recovery path that returns to the earliest part of the process is chosen.

39.3.6 Quality Assurance Frequency Branch

The Quality Assurance and Maintenance function as described in the RAB task analysis includes several different types of functions (Table 13). The RAB task analysis did not give specific frequencies for the performance of all of these QA functions because these frequencies differ between facilities. The frequencies with which these functions are performed has a direct effect on the initiation and catch rates of the QA errors within the model. The number of patients affected by systematic errors depends on the frequency of the QA checks and how many patients are treated between the checks. The model was set to emulate the frequency with which a specific site performs its QA procedures. Personnel from a facility performing RAB treatments were able to provide this information during a site visit.

In the model, the door interlock, radiation warning light, interrupt and abort, source positioning, and time tests are performed once “every clinical use.” A clinical use essentially refers to each time the machine is turned on. If two patients are treated shortly after each other on the same day, then these four QA checks are done only once that day, before the first patient. Source calibration, the auto-radiography test, and the check of the source guide tube and applicator assemblies are done monthly. Source replacement, which includes removing the old source and installing the new source, occurs once every three months.

The model records the total number of treatments that occur, and the frequency of the QA procedures are based on the number of treatments performed per week. For example, if ten treatments per week are scheduled, then using a five-day work week, the model determines that only two treatments are performed per day on average. The QA procedures for each clinical use will occur every two treatments, once per day. The monthly QA procedures, using five weeks per month, will occur every fifty treatments. The QA procedures done every three months will occur every 150 treatments. These frequencies can be modified to represent different facilities.

Parallel Sequences

The majority of the tasks in the RAB process proceed in a linear fashion. However, in two instances portions of the process are done in parallel: monitoring of life support equipment and the treatment set-up.

Monitoring of Life Support. In most cases of bronchial or gynecological RAB treatment, the use of life support equipment is not necessary. In the cases where such equipment is necessary, monitoring is independent from the rest of the process but occurs at the same time. The RAB model represents this parallel process with multiple branching logic. A probability is used to determine how often life support would be necessary. In the network *Patient Preparation*, the process diagram (Figure 32) shows a branch where one path leads to the subnetwork *.1 sched id & track* and another path leads to *.3 life support mon*. The process flow always includes the path that leads to the patient scheduling, identification and tracking subnetwork. The probability of needing life support determines when the process also includes the parallel network of life support monitoring. A Multiple decision type (the M where the paths diverge) is used so that both paths are followed when life support is included.

Network 1 patient prep

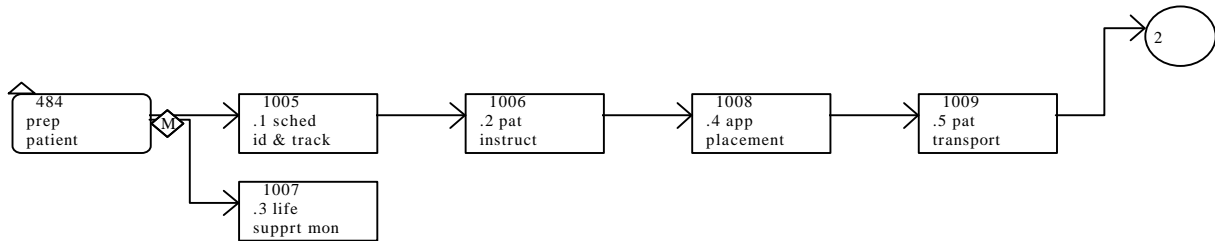


Figure 32. Patient preparation network diagram

Treatment Set-up (inside vs. outside). The other example of a parallel task in the RAB process is setting up the treatment. Different personnel set up the equipment outside the treatment room and prepare the patient inside the treatment room. Many of these tasks occur in parallel. In the network *Treatment Delivery*, the process diagram (Figure 33) shows a branch where one path leads to the subnetwork *.1 trmt stp outsd* (treatment setup outside the treatment room) and one path leads to *.1 trmt stp inside* (treatment setup inside the treatment room). A Multiple decision type is used at the branch point and both paths are always followed. The model continues to execute the subsequent task nodes in parallel until the paths meet when the setup is complete and the treatment is started.

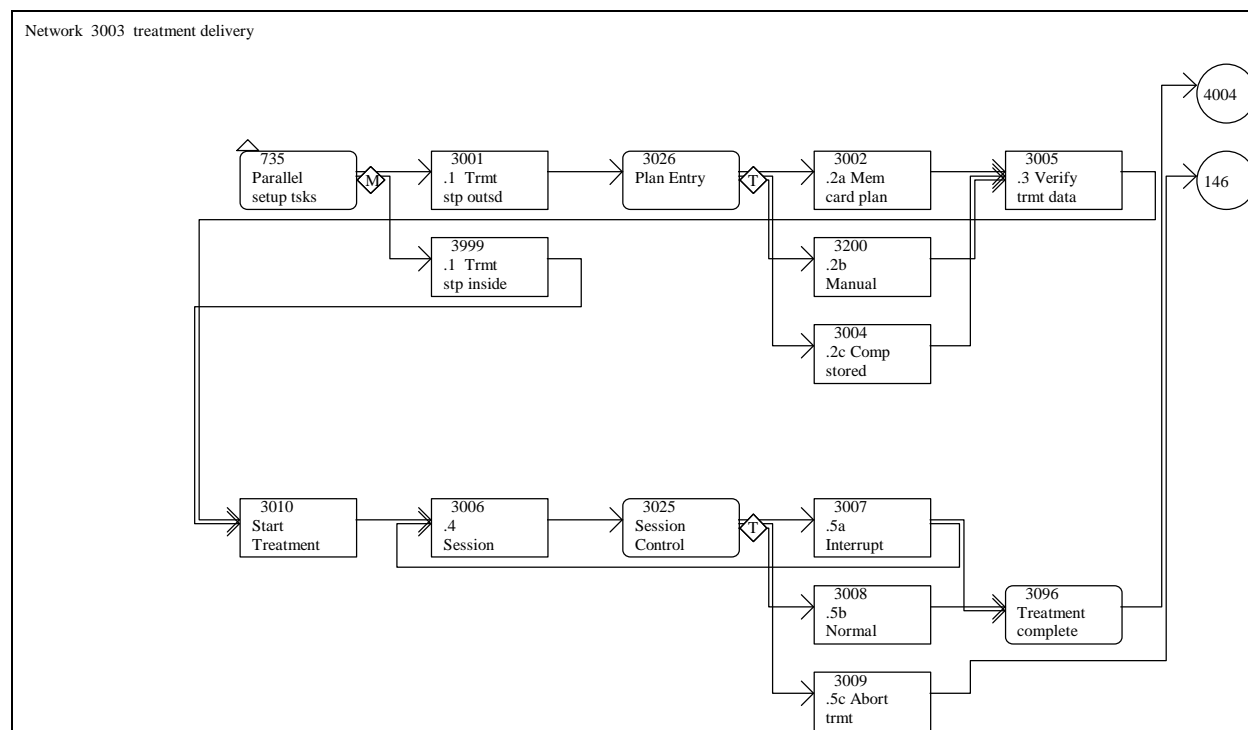


Figure 33. Treatment delivery network diagram

Task Sequence Control

Many times during the RAB process, optional or multiple methods for performing part of the process can occur. Each time one of these points is reached, a decision must be made concerning the method to choose. Examples of path branches occur in the *Treatment Delivery* network and in the *QA* network's *interrupt and abort tests* subnetwork. These are described below.

Treatment delivery. Two branches are in the *Treatment Delivery* network diagram (Figure 33). The first branch discussed follows the *Plan Entry* node and is for the three methods in which treatment plans can be entered into the treatment system. The network shows that the treatment plan can be entered with a card from the treatment planning system (node .2a), manually into the treatment delivery system (node .2b), or by retrieving a stored plan from the memory of the delivery system (node .2c). The method used in the model is determined in the patient profile (see Section 7.3.8). A Tactical decision type (the T at the branching point) is used to determine which of the three paths to follow.

The second branch of this type in Figure 33 is for treatment delivery termination following the *Session Control* node. The three ways that a treatment can be stopped are when it ends normally (node .5b), when it is interrupted (node .5a), or when it is aborted (node .5c). One of the three ways to stop treatment is selected in the *.4 Session monitoring* network (described below).

In the *Session monitoring* network (Figure 34) three systems, the treatment console, the camera, and

the intercom, must be monitored during treatment delivery. Session monitoring tasks are repeated until the time allocated for the session has expired. Each time through the sequence of task nodes a probability determines which of the three systems is being monitored.

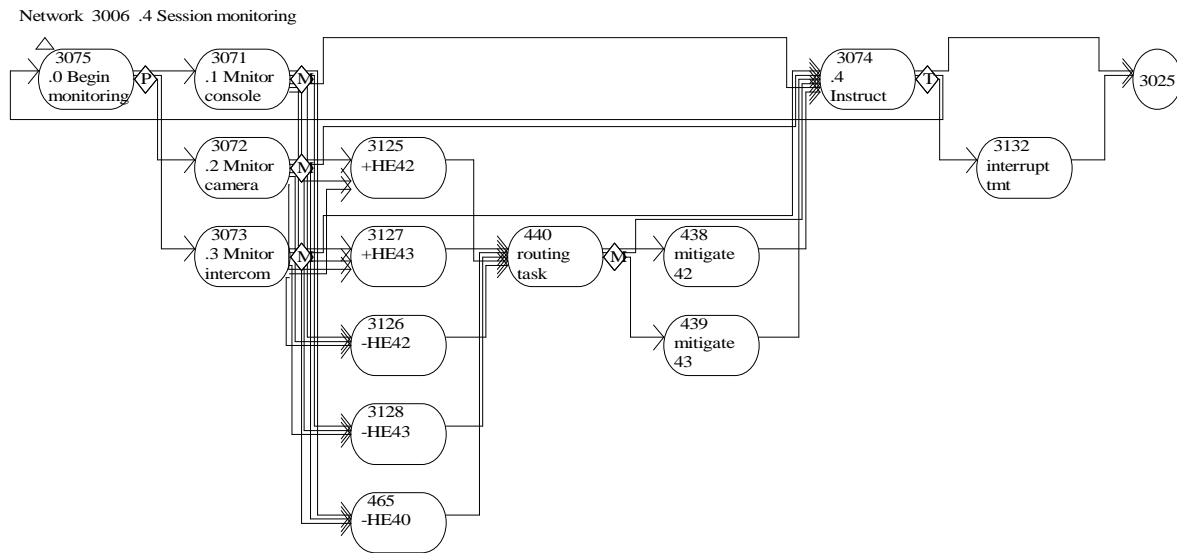


Figure 34. Session monitoring network diagram

Interrupt & abort tests. Three different tests are performed in the *interrupt and abort tests* subnetwork of the QA network. The tasks in this subnetwork are repeated three times, with each of the three tests being performed once.

Selective Task Execution

Most of the tasks in the RAB process are performed for every treatment. However, several parts of the process may not be performed for each treatment or may not always be performed in the same order. One example of this is simulating the treatment using dummy sources. SMEs indicated that simulation is not always done for each fraction. In the *Treatment planning* network a probability determines if the *simulate with dummy sources* subnetwork is executed for a given treatment.

Another example is when a treatment is aborted due to some emergency, such as the source becoming disconnected from the cable. Two sets of tasks may be executed as part of the emergency response and do not occur as part of the normal process. These are disconnecting the source guide tubes and removing the applicators. If the patient must be moved from the treatment room quickly, then the source guide tubes will be removed quickly and the applicators may also be removed. In the subnetwork *Abort treatment*, if either of these tasks is done as part of an emergency, then the subnetworks for them are skipped in the *Post treatment* network.

Repeated Tasks

Small portions of a process can be repeated many times during an RAB treatment. An example is when more than one applicator is used in the treatment and the same process for placement of the

applicator is repeated. The model represents these repeated steps by following a path back to the point where the process to be repeated begins. In this example, the repeat path continues until the total number of required applicators has been placed.

39.3.7 Hardware and Software Scenarios

During this project there was no attempt to represent all the equipment failures that could occur in the RAB process. The hardware failure and software error scenarios are meant to demonstrate one way these types of problems can be represented in a DES. One hardware scenario and one software scenario was selected. The scenarios are built into the model and can be selected by the user to demonstrate how the overall process is affected.

Hardware Failures

The hardware failure scenario simulates an event when the source becomes detached from the cable while it is extended into an applicator. When the model is executed with the hardware scenario set to “On”, the process follows a normal treatment path until the failure occurs during treatment delivery. An error message is recorded while monitoring the treatment session, and the process is sent to the trouble-shooting subnetwork. Once the problem is diagnosed, the process continues through the tasks necessary to abort the treatment session, remove the source from the patient, and remove the patient from the treatment room. This demonstrates the proper discovery and recovery path for the hardware failure. To assess the risk with this or any hardware failure, the user would need to estimate the frequency with which the failure could occur, the probabilities that personnel would fail to correctly perform the steps required to mitigate the problem, and the consequences of failing to mitigate the problem. This demonstrates, in general, one method for how hardware failures can be modeled and how the relative risk of hardware failures can be assessed.

Software Errors

The software error scenario depicts an error in the treatment system software that results in the incorrect calculation of the dwell times. When the model is executed with the software error scenario, the process in the QA network includes the installation of new software through the equipment and software updates subnetwork. The new software contains the error. The process continues until an independent check of the dwell times catches the problem. The model can be modified to include all the places where this error might be caught and the probabilities of failing to catch it.

39.3.8 Model Input

The user can select a variety of data elements to enter into the model prior to execution. These data cause the model to function in many different modes and allows a user to tailor model executions to specific needs. See Appendix G for more detail.

Patient Profile

One set of data that can be used by the model is a Micro Saint programming function called the PATIENT PROFILE. For each patient, this function determines the number of fractional treatments that a patient may receive to complete the prescribed treatment dosage, the number of applicators to use, the method of entering the treatment plan into the treatment system, and the length of the treatment session. As currently set, the function randomly chooses one, two, or three fractions for the patient and one, two, or three applicators to use in each fractional treatment with each possibility

having the same probability. The number of fractions is used to determine how many times the model executes for each patient. These parameters can be changed by the user to more accurately model specific treatment situations. Treatment plans can be entered into the treatment delivery system manually, by transferring the plan from the planning system, or by using a plan stored in the memory of the treatment system. Currently, these methods are distributed equally across the patients. The length of the treatment session is chosen randomly from a uniform distribution of between 10 and 20 minutes and is used in the model to control the session monitoring tasks.

Number of Patients and Number of Treatments Per Week

Two values can be set prior to running the model that have a direct effect on how the model functions. The number of patients determines how many patients are sent through the process. The number of treatments per week can be set to reflect the workload of a specific site but does not equate to the number of patients per week. This is because patients will have more than one treatment fraction and the scheduling for these may spread over several weeks. Therefore, the treatments per week only reflects how many fractions are performed per week. This value is used by the model to determine the frequency of the QA procedures.

Error Initiation and Catch Probabilities

The user can also change the error initiation and error catch probabilities prior to executing the model. The probabilities used by the model are based on two seven-point scales that range from very low (vvl) to very very high (vvh). One scale is for the initiation of errors and the other is for catching errors. Fourteen Micro Saint variables make up the scale, one for each value of each scale.

Error Dependency Switch

The dependencies between errors within the model can be set “On” or “Off” prior to model execution. When the dependency between errors is checked during model execution, the dependency flag is also checked to see if it is “On” or “Off.” The user can compare the results of different model runs to see the difference caused by error dependencies.

Hardware and Software Scenario Switches

Two special scenarios can be used during model execution. The hardware failure scenario and the software error scenario demonstrate how these types of problems can be modeled. The user can turn the scenarios “On” or “Off” before model execution and then run a single patient through the model to see the effect of a hardware failure or a software error.

39.4 Data Output Measures

The following section contains a description of the data that were recorded in the RAB model. Many of the files include redundancies that can be used to cross-check the accuracy of the reporting programs. The first set of outputs contains the data input for the current run of the model (Section 7.4.1). The next set of outputs contains the interim results that are gathered during each fraction while the model is running (Section 7.4.2). The final set of outputs contains the end results of the model run, some gathered at the end of each fraction and some after all the patients have gone through the model (Section 7.4.3). Appendix H includes examples of these files.

39.4.1 Data Input

- General input. This file documents the data input that controls the model execution. It includes the number of patients, the treatments per week, the dependency switch, the hardware and software scenario switches, and the probability values for the initiation and catch scales.
- Patient input. This file documents the model data input from the Patient Profile. For each patient, the file includes the number of fractions, the number of applicators, the session length, and whether the treatment plan was entered manually, transferred from the planning system, or from computer memory.

41.0.1 Interim Results

- Errors initiated. This file documents all the errors that were initiated for the current execution of the model. It includes the patient identification number, the fraction number, the error that was initiated, and the task in which the error was initiated.
- Number of initiations per error. This file records how many times each error was initiated throughout the execution of the model. It includes the error identification number and the number of times that the error was initiated. This file can be used to determine the errors that are initiated most often.
- Number of initiations per task. This file records how many errors are initiated for each task. The file includes the task node number and the number of errors that were initiated at that node through the run of the model. This result can be used to determine the task with the highest number of errors initiated.
- Number of errors per patient. This file documents how many errors were initiated for each patient. It includes the patient identification number and the number of errors initiated for each patient.
- Errors caught. This file documents the errors that were caught during the entire execution of the model. It includes the patient identification number, the fraction number, the identification number of the error that was caught, and the node number in which the error is caught.
- Systematic errors. This file records how many patients were affected by each systematic error. The file includes the identification number for each systematic error and the number of patients that were affected by each error.
- Programmatic errors. This file records how many fractions were affected by each programmatic error. The file includes the identification number for each programmatic error and the number of fractions that were affected by each error.

48.0.1 End Results

- Patient output. This file documents the errors that were initiated and the errors that were caught for each patient. It includes the patient identification number, the fraction number, the identification number for each error that was initiated, and the number of times each error was caught.
- Non-dosage errors. This file documents errors that resulted in non-dosage misadministrations. It includes the patient identification number and the error identification number.

- **Dosage deviations.** This file documents the dosage deviations for dosage errors. The file includes the patient identification number, the fraction number, the error identification number, the error type, and the dosage deviation code.
- **Worst case dosage deviations.** This file reports predicted dosage deviations for each error, even if more than one such error is in the same fraction. The model compares the predicted potential outcome from each error for a specific fraction and reports the single worst case dosage. The file includes the patient identification number, the fraction number, the error identification number, the error type, and the dosage deviation code for the worst case deviation.
- **Execution statistics.** This file documents the total counts of several useful model parameters. It includes the total treatments performed, the total errors initiated, the total errors caught, the total errors that affected patients, the total number of fractions affected by programmatic errors, the total number of patients affected by systematic errors, the total dosage deviations for all fractions, the total dosage deviations for final fractions, the total worst case dosage deviations, and the total worst case dosage deviations for final fractions.

The reports for final fractions refer to final fraction for each patient. In the RAB process, dosage deviations in early fractions can often be “made up” by increasing or decreasing the dosage in subsequent fractions. Sometimes additional fractions beyond those in the original treatment plan can be used to deliver further doses if the original fractions delivered too little. This has led to some controversy concerning whether dosage deviations in early fractions that can be made up in later fractions are considered misadministrations. No attempt was made in the RAB model to “make up” dosage deviations. Instead, it was determined that both dosage deviations for all patient fractions and those for the final fraction be reported. The report of final fraction dosage deviation is simply a restatement of the last fraction for each patient from the dosage deviation report of all fractions. Using these results files the user can see either the dosage deviation of each fraction or, if the assumption that dosage deviations are being made up during subsequent fractions is used, the user can look at only the dosage deviation of the final fraction.

53.1 Verification and Validation

This section documents the two verification and validation (V&V) efforts that were performed as part of the modeling work. The purpose of these V&V efforts was to verify the model structure, check the validity of the model, and assess the degree to which this modeling technology is accepted by potential users. The first effort involved the personnel from PS&EG (Section 7.5.1). The second involved expert RAB practitioners (Section 7.5.2).

53.1.1 Pacific Sciences & Engineering Group

The first V&V involved personnel from the Pacific Sciences and Engineering Group (PS&EG). PS&EG authored the RAB task and error analysis document that was used as a primary information source to develop the model and was in the best position to determine whether the static task analysis was adequately converted into a dynamic simulation model. PS&EG was asked to verify the accuracy of the network flow, to help add the QA function to the model, and to review the errors that were systematic or programmatic. In addition, they were asked to provide input for estimates of probabilities for initiating and catching each human error, describe any dependencies between the

errors, and help develop a method for translating error consequences into treatment results.

PS&EG reviewed the network diagrams and compared them with the NUREG/CR-6125 task and error analyses and their own understanding of the RAB process. They agreed that the majority of the networks accurately represented the information from the task and error analyses in NUREG/CR-6125. Several changes were made to the model based on their recommendations. The majority of these changes included the locations where errors could be initiated and caught. In addition, PS&EG reviewed the assignments of errors to the systematic and programmatic error categories.

Throughout this process, the PS&EG personnel were able to easily understand the network diagrams and the mapping between the diagrams and the information contained in the NUREG/CR-6125.

In addition to reviewing the network diagrams, the personnel from PS&EG reviewed the relative estimates of the probabilities for each human error (Appendix B). PS&EG personnel also described several dependencies between errors in which the initiation of one error can either increase or decrease the likelihood of one or more other errors. These dependencies were incorporated into the model.

Finally, the PS&EG personnel were able to help develop the method for reporting treatment outcomes in the model. They provided input for grouping each of the human errors into categories based on three different types of misadministrations and developed several dosage deviation distributions. The treatment outcomes in the model are based on this mapping between the errors and the types of misadministrations. The details of this method were described in Section 7.3.4.

53.1.2 Expert Remote Afterloading Brachytherapy Practitioners

The second V&V effort involved practitioners who are considered by the NRC to be experts in the field of RAB. The intent was to have this group of SMEs act as a second source of information. Specifically, they were to estimate probabilities for initiating and catching each of the human errors and to estimate dosage deviation distributions for each error type. However, during this visit, the SMEs were unwilling to provide estimates for error probabilities. They stated that any such estimates would be extremely inaccurate due to their own biases and the complexity of the RAB process.

The SMEs believed that this complexity is due to the dynamic nature of the RAB process, which includes differences between facilities, treatment types, and patients. Because of these variants, the errors that can occur may be extremely diverse or have very different probabilities. They stated that the variability of error probabilities is very large.

While reviewing the network diagrams, the SMEs located several instances where the process in the model did not match the general method used at their facilities. They decided that the process described in the RAB task analysis (NUREG/CR-6125, Volume 2), a composite of the processes from several facilities, is not useful for reviewing the process of a specific facility. They described differences in the process, which include variations in the facility, physician, treatment location, and patient. Some of these differences could be addressed using performance shaping factors (PSFs)³ with the error probabilities, but others can be addressed in the task network itself. In addition, they stated that the attempt to combine the two treatment types, bronchial and gynecological, into a single model was not reasonable. They indicated that the differences in the two treatment processes were too great to be successfully analyzed together in the same model.

³IPSFs are factors that affect the probabilities of error such as the level of experience of the personnel, the level of training of personnel, the level of stress, and the level of understanding of process changes, etc.

The group of SMEs felt that effects of PSFs must be understood to help address the variability in the process if either a quantitative or qualitative analysis is done. The PSFs discussed include the following:

- Process Knowledge (experience) - This is affected by new personnel, personnel from other facilities, and new processes.
- Stress
- Training

The effects of these PSFs can be accounted for in several ways. The first is by assigning different numerical values to the seven point probability of error scales. The second is to modify specific probabilities during model execution based on the effect of a specific PSF in the same way that dependencies are modeled.

The SMEs were able to understand the flow of the model from the network diagrams. They agreed that this technique could be very useful and that formally reviewing the process to build the model is valuable. They also concurred that using probabilities in the model would be valuable provided sufficient data existed. However, they do not believe that such empirical data currently exists.

56.1 Using the Remote Afterloading Brachytherapy Model

To use the RAB model to analyze and improve the safety of the RAB treatment process, several procedures should be considered. The first step is to establish baseline values for the RAB model execution (Section 7.6.1). Using these results, an area of focus can be identified (Section 7.6.2), changes can then be made to the model (Section 7.6.3), and their effects can be assessed. (Sections 7.6.4 & 7.6.5).

This section includes the data results from the model execution. This model is not intended to be an accurate predictor of treatment outcomes for a specific facility because it does not represent the RAB process from a specific facility. Additionally, the input data for the model (e.g., error initiation and catch probabilities) were not verified and validated by SMEs. The results are based on four separate executions of the model. Each included 1,000 patients and was set to ten treatments per week. The error initiation probabilities and the use of dependencies were varied across the four executions. The values of the seven-point scale for the probabilities of initiating (P(I)) the errors were lowered from their initial values by a factor of ten for two of the scenarios. The dependencies between the errors were also varied (turned On or Off). Table 17 provides the model settings for each scenario. See Section 7.3.8 for details on setting these values. Examples of the results files are in Appendix H.

Table 17. Settings for the four scenario runs

Scenario #1	Scenario #2	Scenario #3	Scenario #4
Initial P(I)	Initial P(I)	P(I) / 10	P(I) / 10
Dependencies On	Dependencies Off	Dependencies On	Dependencies Off

The error dependencies were set “Off” or “On” during the model runs. This was done to evaluate the effect that error dependencies had on model outcome and to determine whether they should be included in future models of this type.

56.1.1 Establishing a Baseline

Baseline model results are compared to subsequent executions that include changes to the process. Data, such as, the number of errors per patient and the number of total dosage deviations can be used to measure the effectiveness of proposed changes. Based on these measures, goals for improving the safety of the process can also be identified.

The next two tables are examples of the kind of output that can be used for baseline information. Two model runs were made, each with 1,000 patients and error initiation probabilities lowered by a factor of ten ($P(I)/10$). The model was executed with the error dependencies set to “On” and then again with the dependencies set to “Off”. Table 18 and Table 19 show the number of patients who ended up with each level of dosage deviation. Table 18 reports the data for all fractions in each treatment. With the dependencies set “On” or “Off”, the table shows that out of 1,000 patients treated, 13 dosage deviations of greater than 20% over the planned dose were predicted over all the fractions.

Table 19 reports the data for the last fraction of each treatment in which an error occurred that could cause a dosage deviation. With the error dependencies activated, the table shows that out of 1,000 patients treated, 10 dosage deviations of greater than 20% over the planned dose were predicted in the last fraction.

Table 18. Dosage deviations for all fractions

1000 Patients	>20% under	Between 10% & 20% Under	Within 10%	Between 10% & 20% Over	>20% Over
P(I) Initial / 10					
Dependencies = Off	8	13	7	4	13
Dependencies = On	8	8	7	3	13

Table 19. Dosage deviations for last fractions

1000 Patients	>20% under	Between 10% & 20% Under	Within 10%	Between 10% & 20% Over	>20% Over
P(I) Initial / 10					
Dependencies = Off	4	8	4	2	8
Dependencies = On	5	3	3	1	10

56.1.2 Identifying Areas of Focus

The model can be used to determine specific areas of focus for efforts to improve process safety. This is done by identifying the locations in the process in which errors are most likely to cause adverse effects. The results files from the baseline model that contain the number of initiations per error and the number of error initiations per task can be used to determine the most commonly initiated errors and the tasks at which the most errors are being initiated. These errors and tasks will be the highest contributors to the risk of misadministration in the process. Areas of the process that contribute

most to the risk level should receive greater focus during improvement efforts.

The consequences of the errors can be used to help prioritize these focus areas. The dosage deviation and worst case dosage deviation output files report the effect that each error that was not caught had on a treatment outcome. In some cases the most important focus areas will be obvious. Errors that occur frequently, are not caught, and have high consequences are obviously the most important. However, it is possible that some errors may be initiated frequently but have only a minor effect on treatment outcome; others may not occur as often but have very serious effects. In these cases, some trade-offs occur between the error frequency, consequence level, and the proposed changes to the process.

Most Frequently Occurring Errors

The results in the following tables show the errors that occurred most frequently for each of the four scenarios. These data were derived from the output files that recorded the number of initiations of each error.

Table 20 and Table 21 show the most frequently occurring errors and their descriptions for each of the four scenarios. The left most column of Table 20 shows the frequency ranking of each error; number one (1) is the most frequent. The next four columns list the error numbers for each of the four scenarios. The most frequent error for scenarios 1 and 2 was error 48. Table 21 includes the error number, description, and error type for each of the errors listed.

**Table 20. Summary of most frequently occurring errors
(table shows error numbers)**

Ranking	Scenario			
	1	2	3	4
1	48*	48	13	13
2	13	13	48	48
3	8	21	8	1
4	21	47	1	31
5	1	1	31	21
6	2	2	21	3
7		3	2	2
8			3	

* See Table 21 for the error descriptions

Table 21. Descriptions for the most frequently occurring errors

Error Number	Description	Type
48	Damage to applicator	3
13	Applicator moved during transport	4
8	Applicator not placed correctly	4
21	Applicator moved	4

1	Data entry error	varies
2	Patient identification error	non-dosage
3	Patient routing error	1

Tasks Where the Greatest Number of Errors Were Initiated

The results in the following tables list the tasks where the greatest number of errors were initiated for each of the four scenarios. These data were derived from the results files for the number of errors initiated per task. Table 22 and Table 23 show the task numbers and descriptions where the greatest numbers of errors were initiated for each of the four scenarios. The left most column shows the frequency ranking of each task with number one (1) being the most frequent. The next four columns list the task numbers for each of the four scenarios. Task 3.1.7 had the greatest number of errors initiated for scenario 2. Table 23 includes the task numbers and descriptions.

Table 22. Summary of tasks where the most errors were initiated
(table shows task numbers)

Ranking	Scenario			
	1	2	3	4
1	1.5.10*	3.1.7	1.5.10	1.5.10
2	1.4.8	1.5.10	1.4.8	1.4.9
3	3.1.7	1.1.5	3.1.7	3.1.7
4	1.4.9	5.5d.4	4.5.0	1.4.10
5	1.4.10	4.5.0	1.4.9	4.5.0
6	5.5d.4	1.4.10	1.4.10	5.5d.4
7	4.5.0	1.4.9	5.5d.4	1.1.5

* See Table 23 for the task names.

Table 23. Numbers and names for each task where the most errors were initiated

Task Number	Task Name
1.5.10	Move patient from transporter
3.1.7	Connect applicator
1.4.8	Move applicator into position
1.4.9	Secure applicator
1.4.10	Mark applicator
5.5d.4	Record results of QA interrupt & abort test
4.5.0	Perform treatment recording
1.1.5	Track transportation schedule

The top four errors from Table 20 all include problems with applicator placement. The top five tasks from Table 23 also involve applicators. These data results indicate that an initial focus of safety

improvement efforts should be those parts of the process that involve placing and securing applicators.

56.1.3 Proposing Changes to the Process

The next step would be to determine what changes can be made to the process based on the analysis of the model results. While an initial focus for the changes can be identified by the model results, the actual modifications that are made can vary widely. If the goal is to reduce the occurrence of a specific error, the process surrounding the points where that error can be initiated might be altered. If the goal is to prevent the occurrence of a specific error going undetected, then a check might be added. Potential changes will also vary depending on process and resource issues, feasibility issues, facility policies, etc.

Interdisciplinary teams can be used to generate possible process changes. Such teams should include SMEs from the process, as well as personnel familiar with human factors issues, policy issues, and the functioning of the model. The focus points identified by the model should not be used as the only source defining the scope of process reviews. Rather, each focus point should be carefully scrutinized by the team to determine if it is an appropriate point of the process on which to focus safety improvement efforts.

56.1.4 Testing Changes

Once specific process changes are proposed, the model can be used to test how those changes affect the process. The proposed changes can be made in the model and the model can be executed using the same data input as those used for the baseline. The results can then be compared against the baseline results to see the effect the changes had on the process. It is best to make and test only a single discrete change at a time. If multiple changes are made at the same time, it may be difficult to determine which change contributed most to the result. This process becomes iterative as multiple changes are tested and as the focus changes to other parts of the process.

56.1.5 Testing Other Changes

Changes to the process might occur for many other reasons. The purpose of proposed process changes may range from increased efficiency to ancillary administrative requirements. Such changes may be based on specific analyses of the process and may come from sources, such as regulatory or administrative bodies. The effect that these changes have on the safety of the process may not be clearly understood. The DES model can be used to assess whether changes for purposes other than safety have any indirect effect on the safety of the system.

In addition, the cost of implementing process changes can be very large. Using a DES model to test changes before they are implemented can save considerable time and money by helping to limit the process changes to those that, at least, do not reduce safety.

.

57.COMPARING DISCRETE EVENT SIMULATION TO OTHER RISK ASSESSMENT METHODS

This section presents a comparison of traditional PRA techniques with the DES methodology presented in this report. First, there is a short description of the NRC's Risk-Informed Regulation initiative that includes a discussion of how different PRA techniques can be used in support of the regulation (Section 8.1). Following that, there is a short description of the traditional PRA techniques that use fault and event trees and a discussion of how the DES methodology fits into the realm of probabilistic risk assessment (Section 8.2). Finally, these techniques are compared by presenting how each one represents several different process issues common to risk assessment (Section 8.3).

57.1 Risk-Informed Regulation

The NRC is currently developing guidelines and methods for implementing a policy called "Risk-Informed Regulation." For nuclear power plants, these guidelines rely heavily on traditional Probabilistic Risk Assessment (PRA) and Human Reliability Assessment (HRA) methods, most of which are based on fault trees and event trees. This project demonstrates that DES is an alternative risk assessment method that might be more appropriate than the traditional PRA/HRA methods for assessing risks associated with the medical use of radioactive materials.

57.2 Probabilistic Risk Assessment Techniques

This section uses a general definition of PRA to show how both traditional techniques using fault and event trees and the DES methodology are probabilistic risk assessment techniques. An analysis methodology can be said to be a Probabilistic Risk Assessment technique if it is a systematic, analytical technique that answers three questions; 1) what can go wrong, 2) how likely is it (qualitatively or quantitatively), and 3) what are the consequences.

The following are short descriptions of the event tree, fault tree, and DES risk assessment techniques. These are provided to show how each is a technique that can answer the three questions in the above definition. These descriptions are also meant to help the reader understand what aspects of the techniques are being discussed in the comparison and are not meant to provide a detailed explanation of each technique.

Event Trees

Event trees are used to model a specific sequence of events which represent either a process flow or an upset condition within the process. Each sequence that is modeled starts with a single initiating event that is followed by one path for its successful completion or execution and one for its failure. Each branch leads to a subsequent event that also produces a success and failure path. The lowest level in an event tree is a path representing the success or the failure of the process. A probability of success and a probability of failure are used for each branch of each event. The end of each path, either success or failure, will have a probability that is a combination of the values for the whole path. The combination of the probabilities of the failure paths is considered the probability that the process will fail given the initiating event. This probability and the consequences of failure are used to determine the risk in the process.

Fault Trees

Fault trees are used to model the combination of failures necessary to reach an upper level undesired result. The top node in a fault tree is the undesired result. Each subsequent node in the tree is a failure that must occur for the top level fault to occur. Each fault in the tree is decomposed into more detailed components so that the upper fault is expressed as a combination of lower faults within the tree. The decomposition continues until a level is reached where probability of failure data is available for each fault. Boolean logic is used to combine the values from the lowest level of the tree to the top level fault. This value is used as the probability that the top level fault will occur. This probability and the consequences of the undesired result are used to determine the risk in the process.

Discrete Event Simulation

The DES methodology is the technique presented in Section 6 of this report and was used to develop the RAB model presented in Section 7 of this report. It models the task flow of a process including the human errors and equipment failures that could occur. Probabilities are used to initiate errors, failures, and recoveries during model execution. Consequences of errors and failures are used to represent undesired outcomes of the process. In the case of the RAB model, the undesired outcomes were related to radiation dosage. Thus DES is a PRA technique as that term is defined above.

57.3 Comparing Techniques

This section presents the comparison between traditional risk assessment techniques using event and fault trees and the DES methodology. The comparison is made through the methods that each technique employs to express process and analysis issues common to risk assessment.

Data Requirements

All these techniques use similar kinds of data about the process that is being analyzed. Each technique begins with some level of task analysis that provides the necessary understanding of the process to be analyzed. In event or fault tree analysis, this information will be used to develop specific process scenarios or combinations of failures that will be analyzed. The DES methodology uses the task analysis to develop a diagram of the process flow and the errors and failures that can occur during the process. See Section 6.2.1 for information on DES data requirements.

Level of Detail

These techniques all allow a flexible level of process decomposition. Events in event trees can be expressed at any level of detail required to represent the sequence. Faults in fault trees can be continually broken down into component parts until the necessary level of detail is reached. A DES model can also be expressed at varying levels of detail. These levels of detail can be organized using the networks and subnetworks of the network diagram. See Section 6.2.1 for more information on level of detail in DES.

Errors without Recoveries

In building a DES model, the analyst identifies the places where errors can be initiated and the place where they can be caught. After doing this, the analyst can, by inspecting the model, identify those errors that have a place of initiation, but have no place where they can be caught. If the model is an

accurate representation of the process, the licensee might want to redesign the procedure to add a check point to catch such errors. See Section 6.2.1 for more information on identifying errors that cannot be caught within the process.

Repeated Parts of the Process

In the RAB process, a series of tasks may be repeated during a treatment. For example, if more than one applicator is being used, the tasks involved with placing the applicator in the patient will be repeated for each additional applicator. The potential exists for errors to be initiated or caught each time the process is repeated. Fault and event trees model this by changing the failure probabilities to reflect the multiple potential for failure or recovery. DES models this more straightforwardly, the sequence of tasks would be repeated allowing errors to be initiated or caught during the task execution. See Section 6.3.3 for more information on branching logic and repeating tasks.

Recoveries from Errors and Failures

Recovery from errors and failures can be expressed in fault and event trees. In a fault tree, any given fault can be expressed in the tree as the combination of the fault itself and failing to recover from the fault. The fault and the failure of the recovery will have probabilities that can be combined to express the probability of the upper level fault. In an event tree, recoveries are often expressed with a path to another part of the sequence.

When an error or failure is caught in a DES model, it is assumed that the error is recovered, and part of the process is repeated (section 6.3.4). In a medical process, for example, if an error in the treatment plan is caught, part of the treatment planning process may be repeated. Repeating parts of the process can result in additional error initiations and catches. The DES methodology explicitly represents these recovery paths.

Dependencies

Dependencies can be expressed in the same way for all these methods. Each error probability is altered based on the effects of the dependency. In a DES model, dependencies are built into each error probability calculation (Section 6.5). This allows the probabilities of the errors to change dynamically during model execution depending on the initiation or catch of other errors within the model.

Programmatic and Systematic Errors

The RAB DES model is able to represent the treatments of multiple patients by continuously executing the model for each patient. This makes it possible to represent the effect of programmatic and systematic errors (Section 6.10.6). Programmatic errors in the RAB process are defined as those errors that can affect more than one treatment fraction of a single patient. In the RAB DES model, a flag is set when an error is initiated. The flag is reset if the error is caught or, for most error types, before the next treatment fraction begins. If the error is programmatic and is not caught, then the error flag is only reset if a new patient is being treated. The error will still be able to affect subsequent treatment fractions of the same patient. Systematic errors are those that can affect the treatments of multiple patients. In the RAB DES model, the flags for the initiation of systematic errors are only reset if the error gets caught. In this way, the model can express how the error can affect multiple patients. Like failures in repeated parts of the process, fault and event trees can only

express the effects of programmatic and systematic errors by altering the probability of specific faults or events. DES models it much more straightforwardly and automatically accounts for repetitions in the process.

Performance Shaping Factors

PSFs, such as, experience, stress, and training can be expressed in the same way for all these methods. The probabilities of error or failures can be adjusted depending on how they are affected by the PSF. In fault and event trees, the analyst must alter the values of each probability in the tree that is affected by a specific PSF. Although this was not done for the RAB model, in a DES model the probabilities can be made functions of the PSFs, and the PSFs can be variables and have changing values. The PSF could be set by the analyst at the beginning of the model execution, or it could be calculated as a function of the events in the scenario, depending on the desired analysis.

Determining Areas of Focus

One of the purposes of risk analysis is to determine what parts of a process contribute most to the level of risk. Fault tree analysis uses importance measures within the tree logic to determine the relative importance of each basic failure within the fault tree. This is done by determining the effect that each of the basic events has on the frequencies of the various outcomes. Likewise, individual events in event trees can be assessed for their relative contribution to the frequency of various outcomes. In the RAB model, the areas of focus are determined by identifying the most frequently occurring errors and the tasks where the most errors occur (Section 6.11).

Consequences

The regulatory community defines risk as the probability of an incident combined with its consequences. The risk involved in a process or system increases with the probability of the error or sequence of errors and the consequence of the error(s). Errors that have a very high probability but very little consequence do not represent a serious risk. Likewise, errors with serious consequences do not represent a serious risk if they have a very low probability of occurrence. An example in a context where PRA is commonly used is the probability of errors and/or failures in a nuclear power plant that result in the loss of coolant pressure combined with the consequences of that loss. The consequences can be represented in terms such as damage to the facility or potential harm to the public.

The RAB model predicts the frequency with which each error occurs across multiple treatments (Section 7.4.2). However, the representation of consequence was limited to what is currently acceptable within the medical context. In the context of RAB treatments, it is not possible to say that an error or combination of errors will result in specific harm to the patient. As such, it is difficult to describe the consequence of the errors. Instead, the RAB model uses the consequences currently accepted by the regulatory community. These are known as ‘misadministrations’ (Section 7.3.4) and are defined by the amount of deviation in radiation dosage, not the effect to the patient. This should be viewed as a context-driven limitation of the RAB model and not as a limitation of the DES risk assessment methodology.

Monte Carlo Techniques

Monte Carlo techniques are used for uncertainty analysis for fault trees. The variability in the probability of the top level fault is assessed based on the uncertainty of the basic events. Monte Carlo

technique is used as different values for basic events are randomly selected from their uncertainty distributions and the probability of the top level fault is recalculated. In DES, model results are based on numerous executions in which errors occur based on their distributions and Monte Carlo sampling. The uncertainty in the results can be assessed by simulating a large number of treatments.

Presentation

Throughout the risk assessment process, analysts are often required to present or explain the analysis technique. This is especially true during verification and validation efforts when analysts work with SMEs to assess the accuracy of trees and models. Although this is a very subjective issue, fault trees can often be difficult to describe to people who are not familiar with the technique. In addition, it can be difficult for SMEs to determine if the analysis is complete when reviewing specific fault scenarios. Event trees are a little easier to understand because they follow the flow of the process from an initial point. However, the completeness issue is still difficult because typically an event tree models only what happens after a specific point and not the entire process. The network diagram of a DES may be easier to understand (Section 7.5). This is because it is a natural mapping of process steps, and the error initiations and catches are represented at the points in the process where they can occur. The issue of completeness can still be difficult because SMEs must determine if all the possible errors and failures that could adversely affect the process have been identified and modeled correctly. It may also be easier to adopt a DES model across sites with slightly different processes than it would be to make changes in fault or event trees. Because the network represents a natural map of the process, it may be easier to adapt a model to other sites by tracing the network flow.

Complementary Techniques

Traditional risk analysis for power plants often begins with a focus on the equipment failures and not on human actions. This is largely due to the automated nature of normal power plant operations. The processes are not usually task-oriented. Human actions are often responses to equipment issues or upset conditions that may occur during plant operations. Thus, the analyst is interested in what faults or events can interrupt the plant as it is running. This is why fault and event tree analysis is so often used to analyze normal power plant operations. The process can either be analyzed from the point of an initial event or from the point of the sequence of faults that must occur to produce an undesired result.

In medical processes, such as RAB, there is little automation. The majority of the treatment process involves sequences of human actions. The focus is on errors and failures occurring during the flow of the process rather than upset conditions. The DES methodology is uniquely suited for such processes because of the way it models the process flow. Power plant operations that follow a sequence of actions or a procedure, such as, start-up or maintenance may also be more suited to analysis using the DES methodology.

Likewise, in the RAB process, there is a point where traditional fault or event tree analysis may be more appropriate. The treatment delivery itself is an automated process under the control of the treatment computer. Like normal power plant operation, the human personnel are simply monitoring the treatment for problems. In the RAB DES model, a loop was created that cycled for the duration of the treatment delivery. Within that cycle, there were opportunities for errors to be initiated and caught. It may be more effective to model this part of the treatment process with traditional fault or event tree analysis

Acceptance

Fault and event trees are part of a widely accepted risk analysis process. The software that is used has been in development for a number of years and is also well accepted (NUREG/CR-6116). Likewise, DES is a well accepted technique for modeling processes for a host of different types of analyses, such as, human performance, resource utilization, efficiency, cost, scheduling, and training (Section 3). The Micro Saint software (Section 5) that was used to develop the RAB model is a well accepted tool for DES.

Ease of Use and Speed

All of these techniques are complicated and involved. They require the resources of analysts who are very familiar with risk assessment and working with SMEs. They must also understand the techniques and software that will be used.

The software that is used to create process simulations is often thought to be slow when large numbers of model executions are involved. For the purposes of this project, the RAB model was executed for 1000 patients. That is, the model cycled 1000 times. This took approximately 20 minutes using a computer with a 133 MHz processor.

58.RESULTS

The two primary objectives of the project documented in this report were (1) to determine the feasibility of using Discrete Event Simulation (DES) modeling to assess the effects of human errors and hardware and software failures on process safety within RAB in particular and within medical processes that use radioactive materials in general, and (2) to compare DES with other risk assessment methods for evaluating medical processes that use radioactive materials. The model represents a hypothetical RAB process flow based primarily on the task and error analyses from NUREG/CR-6125, the use of the MicroSelectron HDR equipment for bronchial and gynecological treatments, and estimates of human error probabilities and error dependencies for this hypothetical process. The hardware and software failures included in the model are not meant to be a thorough treatment of all possible equipment failures within the RAB process. Rather, they are included to show how equipment failures and their effects on the process can be represented in a DES model. This model is not intended to be an accurate predictor of treatment outcomes for a specific facility because it does not represent the RAB process from a specific facility. Additionally, the input data for the model (e.g., error initiation and catch probabilities) were not verified and validated by SMEs.

The RAB process was described in NUREG/CR-6125 as a structured and layered series of discrete events. These events were modeled using Micro Saint modeling software. A methodology was developed for adding the initiations and catches of human errors and equipment failures to the model. This methodology allows the use of probability distributions for initiating and catching the errors. The model can report the numbers and types of errors that could occur in the process and the events that lead to potential misadministrations. An advantage of this technique is that the model will report misadministrations that occur regardless of whether they could be detected in the real process.

Two separate verification and validation efforts were performed for this project. In both cases, the techniques used in the model were easy to communicate and were well understood by the subject matter experts. The SMEs also indicated their support for DES as a useful way to represent the RAB treatment process. The finding at the second review was that the processes for performing RAB treatments were considerably different at each facility and also varied greatly from one treatment type to another.

The RAB model simulation was executed four times with varying error initiation probabilities and use of dependencies. Each model execution included 1,000 patients. The most frequently occurring errors and associated tasks were identified for each of the four scenarios. Possibly the most interesting result from the four model executions was that the same set of the most frequently occurring errors appeared in all four scenarios. In addition, the same tasks were responsible for initiating the highest number of errors that resulted in incorrect treatments. The results of the simulation indicate that the relative risk of incorrect treatment in the RAB process can be identified using relative estimates of error probabilities. DES can be used to assess the locations at which errors are most likely to occur, and the points in the process where errors can be identified and corrected.

The results also indicated that the dependencies included in the model did not significantly affect the process outcome. However, the dependencies included in the model did not represent a complete set of dependencies with the RAB process. Therefore, it is not possible to conclude that dependencies should not be included in models of this type.

The process of developing and testing any proposed changes to the process was not performed during this project. However, a distinct difference between the treatment process in the RAB model and that in the specific sites visited during the project was noted. In the RAB model, the patient is in one

location while the applicators are placed and then moved to another for the treatment. This is the process represented in NUREG/CR-6125. Several of the most frequently occurring errors have to do with movement of the applicator during transportation. In each of the sites that were visited, the patient was not moved between the placing of the applicators and the actual treatment delivery. This helps to prevent the movement of the applicators problem noted in NUREG/CR-6125 and the RAB model. It is not known whether the process at these sites once involved this transportation step. However, it is important to note that this point of focus, predicted by the RAB model, has a workable solution that is currently implemented at some sites.

At the time of this project, the NRC was very interested in misadministrations in medical processes, such as RAB. We chose not to try to translate misadministrations into consequences for patients or staff. The potential adverse effects of radiation doses (other than extreme cases of radiation overdose) is a matter of medical judgment. The number of misadministrations predicted by the RAB model were only used as a measure of problems and improvements. There are many other model outputs that can be obtained from the RAB model and from DES models in general and, even without reporting misadministrations, the process of analyzing frequent errors and tasks where the most errors occur is still important.

58.1 Conclusions

DES is a well established technique for simulating processes that can be described by discrete events, such as tasks performed by humans. Many tasks performed by machines and by computer hardware and software are discrete events that can be modeled by DES. It is generally contrasted with continuous event simulation. These generally rely on continuous mathematical equations that are reevaluated at very short time intervals.

The process of building the network diagram for a DES from a description of tasks and determining the process flow is straightforward and understandable. Models of this type have been used successfully to analyze aspects of processes, such as, human performance, resource utilization, efficiency, cost, scheduling, and training. This project has demonstrated that DES can also be used to assess human error and human responses to hardware and software failures in the RAB treatment process. Equipment failures could also be included in a model that use on-demand failure probabilities and error nodes much like the human errors.

Results of the model can be used to answer the “what if” questions of proposed changes to increase process safety. A proposed change to the process can be modeled. The effects of the change can be assessed by comparing model results from before and after the change. Using the model to assess the effectiveness of changes can save time and money by helping to limit changes to those that increase, or at least do not decrease, the process safety. Using models to assess process changes has proven to be cost effective. The cost of developing a medical process model and using it to test proposed process changes can be more cost effective than making a change to an actual process and finding out it does not provide the expected result.

The methodology for building models and using DES for risk assessment can be generalized to not only medical processes similar to RAB treatments, but also to any process for which the required information is available. If the process can be described by discrete events, it can be modeled using a network diagram. Human errors and equipment failures that can affect the system can be included in the model. Probabilities can be used to simulate the initiations and catches of errors and failures, and associated consequences can be used to assess the importance of errors and failures and predict

process outcome.

DES is also uniquely qualified as an alternative and complementary risk assessment method for the NRC's policy of "Risk-Informed Regulation." This policy may be used when implementing changes to the licensing bases for operations over which the NRC has regulatory duties. The ability of DES to assess the effects of changes to the process being modeled can be used very effectively towards the goals of this policy.

The one main advantage that DES has over other risk assessment techniques is that it models both events that are errors and failures and events that occur when the process is proceeding correctly. In this sense, the entire process is included in the model. The complex interactions between events, such as, repeated tasks and errors that affect more than one treatment or patient, occur as part of the execution of the model, thus reducing the chances that some of the interactions will be missed. The SMEs who were involved in this study were easily able to understand the network diagram used in the RAB model. This has made the process of validating the flow of the model more straightforward. DES is limited, as are most risk assessment methods, by the lack of empirical data for human error probabilities.

The process of building a DES model can be complicated. Like the other risk assessment techniques discussed in this report, building a DES model requires an analyst who is familiar with the modeling techniques. In this respect, there is no advantage or disadvantage in using DES modeling over other risk assessment techniques.

This study also showed that a general RAB model is probably not sufficient for analyzing the process at a specific facility. The RAB model developed for this project represents a composite process of several different facilities and has been judged to be too general for use in assessing a specific site.

58.2 Recommendations

This section presents recommendations based on the results of this project and the belief that continued work in this area will be beneficial to the arena of risk assessment as a whole. The first recommendation is for the development of a site-specific model to assess and refine this methodology. The second recommendation includes further development of the use of PSFs and dependencies, possibly combined with other DES capabilities. Finally, a customized software tool could be developed for the purpose of assessing risk for medical processes using the DES methodology.

58.2.1 Site-Specific Model

The next step in assessing the utility of DES as a risk assessment technique should be to develop a model for a specific medical treatment type at a specific facility. The current RAB model can be tailored to the process flow and errors for a specific treatment and facility can be developed. In this way, the model and the DES methodology could be tested in a more realistic setting. The project should include using the model to determine areas of the process for which changes can be proposed to increase safety. The model can then be used to assess the effectiveness of the proposed changes.

58.2.2 PSFs and Dependencies

The current RAB model makes limited use of dependencies and does not model performance shaping factors, such as, training, stress, experience, etc. Further study on how a more complete set of dependencies may affect the outcome of model execution is warranted. In addition, a methodology for including PSFs in the model could be developed in future work. Using performance shaping

factors, such as experience would allow the analyst to compare model results across different levels of personnel experience.

DES models are often used to assess workload in terms of time limitations and resource constraints, such as personnel. High workload is a contributor to stress. Future work with this technique might include adding workload analysis into the RAB model that includes patient schedules and personnel availability. The workload in the process could then be used to estimate the stress on personnel and change the error probabilities dynamically within the model.

58.2.3 Software Tool

A software tool can be developed specifically for the purpose of using DES to assess the human errors and system failures within medical processes that use radioactive source materials. The tool would consist of the Micro Saint modeling engine and an easy to use interface. The interface would provide the tools needed for personnel to model the desired process with a minimum of training and little or no background in DES modeling techniques. In the same way that specific software is currently used for traditional PRA/HRA, the model would include types of data analyses specific to using DES for risk assessment.

59. REFERENCES

Callan, J.R., Kelly, R.T., Quinn, M.L., Gwynne, J.W., Moore, R.A., Muckler, F.A., Dasumovic, J., Saunders, W.M., Lepage, R.P., Chin, E., Schoenfeld, I., and Serig, D.I., "Human Factors Evaluation of Remote Afterloading Brachytherapy," NUREG/CR-6125, Vol 1-3. Pacific Science, & Engineering Group, 1995.

Code of Federal Regulations, "Medical Use of Byproduct Material (10 CFR Part 35, Subpart A, Section 35.2)," January 1, 1994.

Federal Register, "Use of Probabilistic Risk Assessment Methods in Nuclear Regulatory Activities; Final Policy Statement," Vol. 60, No. 158, August 16, 1995.

Jones, E.D., Banks, W.W., Altenbach, T.J., Fischer, L.E., "Relative Risk Analysis in Regulating the Use of Radiation-Emitting Medical Devices," NUREG/CR-6323, Lawrence Livermore National Laboratory, 1995.

Kirwan, B., and Ainsworth, L. K., eds., *A Guide To Task Analysis*, Taylor & Francis, London, Washington, DC, 1992.

Meyer, M. A. & Booker, J. M., "Eliciting and Analyzing Expert Judgment," NUREG/CR-5424, Los Alamos National Laboratory, January 1990.

NUREG-0090, "Reports to Congress on Abnormal Occurrences," October - December 1992.

Russel, K. D., Atwood, C. L., Galyean, W. J., Sattison, M. B., Rasmuson, D. M., "Systems Analysis Programs for Hands-on Intergrated Reliability Evaluations (SAPHIRE) Version 5.0," NUREG/CR-6116, EGG-2716, Idaho National Engineering Laboratory, December 1993.

Tortorelli, J. P., eds., "A Workshop on Developing Risk Assessment Methods for Medical Use of Radioactive Material," NUREG/CR-0144, Vol. 1, Idaho National Engineering Laboratory, August 1995.